Lotus

# Enterprise Integrator

3.1
RELEASE

User Guide

Lotus

# Enterprise Integrator

3.1

RELEASE

User Guide

# Contents

# Chapter 1 Introduction

This chapter provides information about the organization of this manual and terms you should know when working with Lotus Enterprise Integrator (LEI). This chapter also includes an overview of LEI, including a description of its features, functionality and architecture.

## Organization of this Manual

This manual includes the sections described in the following table.

| *Section* | *Description* |
|---|---|
| Chapter 1<br>Introduction | This chapter provides information about the organization of this manual and terms you should know when working with Lotus Enterprise Integrator. This chapter also includes an introduction to LEI, including an overview of the architecture. It also states how to contact Lotus. |
| Chapter 2<br>Starting and Running the LEI Server | This chapter provides information about starting the Lotus Enterprise Integrator server and using the server commands. Also included are instructions for configuring the LEI server to autostart at system startup. |
| Chapter 3<br>LEI Development Client | This chapter provides information about the LEI Development Client. |
| Chapter 4<br>LEI Administrator | This chapter provides an introduction to the LEI Administrator and information on using it. |
| Chapter 5<br>Introduction to Connectors | This chapter gives an introduction to Connectors to data sources and information about Connector options. |
| Chapters 6 – 15<br>LEI Connectors | These chapters describe each LEI base Connector. |

*continued*

| *Section* | *Description* |
|---|---|
| Chapter 16<br>MetaConnectors | This chapter provides information about MetaConnectors and how to create and use them. |
| Chapter 17<br>Introduction to LEI Activities | This chapter introduces LEI Activities. It includes descriptions of items that are common to all Activities, and provides basic information for working with Activity documents. |
| Chapters 18 – 28<br>LEI Activities | These chapters each describe a specific LEI Activity. |
| Appendix A<br>Lotus Enterprise Integrator and Data | This appendix provides information about how LEI maps data types between different applications. |
| Appendix B<br>Common Gateway Interface for LEI | This appendix provides information about using the Lotus Enterprise Integrator CGI. |
| Appendix C<br>Character Sets | This appendix provides a complete list of the character sets. |
| Appendix D<br>Error Messages and Troubleshooting | This appendix provides information for troubleshooting LEI. Included is a list of error messages, reasons for the errors, and descriptions of solutions. |
| Appendix E<br>Tutorials | This appendix provides tutorials for creating LEI Activities. |

## Related Documentation

### Lotus Enterprise Integrator Documentation

For more information about Lotus Enterprise Integrator and related products, refer to the following documents:

- *Lotus Enterprise Integrator Release Notes* – The release notes (readme.txt) contain information about the current release of Lotus Enterprise Integrator that may not be included in the printed documentation. It's a good idea to read the release notes to review any updated information they may contain.
- *LEI Domino Connectivity and Installation Guide* – This manual provides Lotus Enterprise Integrator installation and configuration instructions. It also provides information on how to set up Domino Connectors, including information about required software and instructions for testing connectivity.

The *LEI Domino Connectivity and Installation Guide* comprises what was previously two separate but related books – *Domino Connector Connectivity Guide* and *Lotus Enterprise Integrator Installation Guide.* This manual ships as a single book but as two separate .nsf databases (lccon.nsf and leiig.nsf).

- Up-to-the-minute additional information about LEI can also be found at the following Web sites:

  `www.lotus.com/dominoei`

  `www.lotus.com/developers`

## Other Documentation

For more information that you may find helpful, refer to the following documents:

- *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* – This manual describes the LotusScript Extensions for Domino Connectors, which can be used in writing scripted sessions for accessing enterprise data (lsxlc.nsf).
- *Lotus Enterprise Integrator Domino Connector Java Class Library Reference Guide* – The Java LC classes and remote console are not included with this distribution, however they are available for download. You can locate them by choosing the "Lotus Connector Classes for Java" link from the www.lotus.com/dominoei main page.
- *Domino Administrator's Guide* – Provides information for configuring and administering a Notes installation.
- *LotusScript Language Reference* – Provides information about writing LotusScript programs. This could be useful if you want to use the LEI LotusScript Extensions (LSX) to write custom Activities.
- *Additional Domino Connector Documentation* – Lotus Development sells additional Domino Connectors for enterprise systems including Enterprise Resource Planning (ERP) and Transaction Processing Systems. Specific documentation about those Connectors is included with the Connector software and package. You may need documentation for the specific databases, ERP, or transaction processing systems that you are using.

## Contact and Support Information

Lotus provides extensive support for its products. The following sections describe the different ways in which you can get help on using Lotus Enterprise Integrator (LEI), as well as information on how to contact us with suggestions and recommendations.

LEI 3.1 is certified for Domino and Notes Release 5.03 and 4.67 and greater. LEI 3.1 is certified on the following platforms - Window NT 4.0 with Service pack 6a, Sun Solaris 2.7, and IBM AIX 4.3.3.

### The Lotus Enterprise Integrator Web Site and Developers Forum

To keep up-to-date with the latest discussions, technical notes, and updates, visit the Lotus Enterprise Integrator Web sites listed below:

`www.lotus.com/dominoei`

`www.lotus.com/developers`

### Lotus Technical Support

You can reach the Lotus Enterprise Integrator Technical Support group at:

1-800-346-6388

### Contacting Third Party Support

In some cases, you may want to contact Customer Support for issues with a third-party application used in the LEI environment. These could include databases you may be using, such as Sybase or Oracle. Refer to the documentation for the specific product for more information.

## Getting Started with Lotus Enterprise Integrator

The steps below summarize how to get started with Lotus Enterprise Integrator.

1. Install Lotus Enterprise Integrator - See the *LEI Domino Connectivity and Installation Guide*.
2. Establish and Test Database Connectivity - See the *LEI Domino Connectivity and Installation Guide*.
3. Start the LEI Server - See Chapter 2 of this manual.
4. Build Connectors for each external data source. See the chapter that describes the type of Connector you want to create.

5. Build Activities - See the chapter that describes the type of Activity you want to create.
6. Schedule and Run Activities - See the chapter that describes the type of Activity you want to create. Also refer to Chapter 18, "Introduction to LEI Activities."
7. In case of difficulties, refer to the Troubleshooting appendix.
8. For practice examples, refer to the Tutorials appendix.

## Introduction to Lotus Enterprise Integrator

Lotus Enterprise Integrator is an enterprise integration tool that moves data between disparate external data sources.

### Lotus Enterprise Integrator Architecture

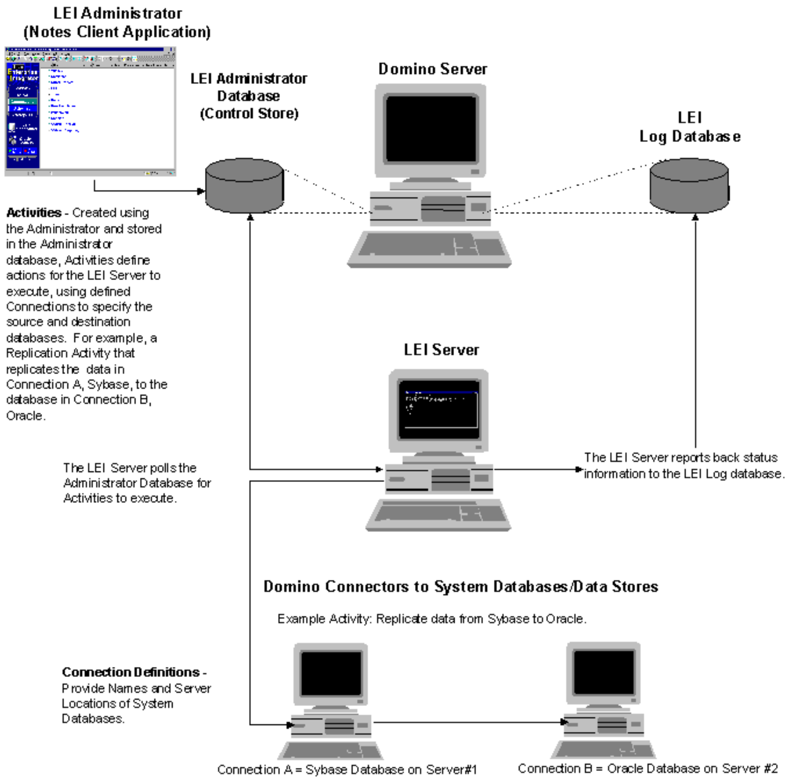
LEI is a data distribution application designed to enable enterprise-wide data access across multiple platforms. It is comprised of three main components:

- Lotus Enterprise Integrator Server
- Lotus Enterprise Integrator Development Client
- Lotus Enterprise Integrator Administrator

#### Overview

The figure below illustrates the components of the LEI architecture and shows how data is processed by LEI. The LEI Administrator is used to administer the configuration and operational elements of the LEI system, including Connections, MetaConnections, and Activities. The LEI server periodically polls the LEI Administrator database for Activities to execute. When it finds an Activity that is scheduled to run, it executes that Activity. An Activity defines the actions that the LEI server will perform, such as transfer of data, replication of data, file system commands and so on.

Each component is described in more detail in the sections below.


LEI Administrator
(Notes Client Application)
LEI Administrator Database (Control Store)
Domino Server
LEI Log Database
Activities - Created using the Administrator and stored in the Administrator database, Activities define actions for the LEI Server to execute, using defined Connections to specify the source and destination databases. For example, a Replication Activity that replicates the data in Connection A, Sybase, to the database in Connection B, Oracle.
LEI Server
The LEI Server polls the Administrator Database for Activities to execute.
The LEI Server reports back status information to the LEI Log database.
Domino Connectors to System Databases/Data Stores
Example Activity: Replicate data from Sybase to Oracle.
Connection Definitions - Provide Names and Server Locations of System Databases.
Connection A = Sybase Database on Server#1
Connection B = Oracle Database on Server #2

## LEI Server

The LEI Server is an engine that polls the LEI Administrator database (also referred to as the Control Store) for instructions to execute. These instructions are in the form of LEI Activities.

Activities can be declarative or scripted. Declarative Activities are forms-based Activities created with the LEI Administrator. Scripted Activities are written in LotusScript, using the Domino Connector LSX with LEI LotusScript Extensions. Once an Activity is written, it can be run from the Administrator or from the operating system command line.

The LEI Server runs as a multi-threaded, multi-processing task on server operating systems. It performs the actual work in transferring data between external sources, such as between Oracle and Sybase or between SAP and Notes, etc.

The LEI Server is described in Chapter 2 of this manual.

### LEI Development Client

The LEI Development Client is an installation option that is provided as a development tool. It enables the development and testing of Activities and Connection without requiring a full LEI Server installation. It's a useful tool for doing early system setup and testing before deployment.

The LEI Development Client also provides action buttons for interactive browsing and field mapping of data with the LEI Administrator.

The LEI Development Client is supported on Windows only.

The LEI Development Client is described in detail in Chapter 3 of this manual.

### LEI Administrator

The LEI Administrator is a Notes application that enables users to create Connections and Activities. It uses a variety of forms and documents, including:

- Configuration
- Connections
- MetaConnections
- Activities

See subsequent sections for more information about each of these items.

The LEI Administrator is described in detail in Chapter 4 of this manual.

## LEI Connections

LEI Connections provide connectivity to external data sources. Connections specify the names and network locations of each source. Use the LEI Administrator form specific to the type of Connection you want to build. LEI provides Domino Connectors for the following data sources:

- Lotus Notes
- DB2
- EDA/SQL
- File System
- Open Database Connectivity (ODBC)
- Oracle 7
- Oracle 8
- OLE DB

- Sybase
- Text

Before building Connections, verify connectivity using the communications software appropriate to the specific data source. For more information about establishing and testing connectivity, refer to the *LEI Domino Connectivity and Installation Guide*.

Connection Options enable you to modify operational aspects of specific Connections. They appear as strings within the Connection document.

For more information about building Connections, refer to Chapter 5 of this manual.

### LEI MetaConnections

A MetaConnection is a special LEI Connection. A MetaConnection "wraps around" a base Connection, appearing to the Activity like the base Connection but providing additional functionality. LEI provides the following MetaConnectors:

- Order: The Order MetaConnector can be used to obtain a consistent ordering of results. For example, dictating a common ordering scheme between EBCDIC and ASCII server data, resulting in accurate replication and data evaluation operations. The Order MetaConnection orders the result set based on user-defined criteria.
- Collapse/Expand: The Collapse/Expand MetaConnector collapses multiple records with a common key value into a single record with multi-value fields, or expands multi-value fields into multiple records. It can be used to transfer multiple source records to a single target record, or vice versa. It collapses records on fetch operations, and expands them on write operations according to user-defined criteria.
- Meter: The Meter MetaConnector measures data access rates and volumes and provides other statistical information.
- Connection Broker: The Connection Broker MetaConnection allows you to attach end-user authentication to your data sources.
- Trace: The Trace MetaConnector enables you to capture trace data. Options exist for specifying a timestamp, where to capture trace data, and what file name to use for logging output. This MetaConnection is usually used in a Customer Support-assisted situation.

See Chapter 16 for more information about MetaConnectors.

## LEI Activities

Activities are the core of LEI. An LEI Activity tells the LEI server what to do, and when to do it. Activities are defined using the LEI Administrator.

Specific types of LEI Activities include:

- Admin-Backup Activity - Provides a way to schedule and execute regular backups of your LEI Administrator databases.
- Admin-Purge Logs Activity - Provides a way to schedule and execute the purging of LEI logs.
- Archive Activity - Provides a way to schedule and execute regular archiving of your databases.
- Command Activity - Enables execution of operating system and database commands.
- Direct Transfer Activity - Allows you to transfer data between a source and target database by specifying SQL queries, Notes formulas or other data source-specific commands.
- Java Activity - Provides a way to schedule and execute Java programs with the LEI server.
- Polling Activity - Allows you to monitor data sources for specified conditions. When a condition is met, LEI immediately executes one or more specified Activities.
- RealTime Notes Activity - Enables a Notes form for real-time interactive access with back-end data.
- Replication Activity - Enables data synchronization between dissimilar databases.
- Scripted Activity - In addition to the standard declarative Activities that can be defined using the LEI Administrator, LEI also provides LotusScript Extensions that can be used to create scripted Activities. This enables you to develop more advanced Activities that can perform data manipulation and data massaging. For more information, see the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide*.

Each Activity is described in its own chapter in this user guide. Refer to the chapter for the specific Activity for details on the creation and operation of the Activity.

## LEI Logs

LEI maintains a system log database that tracks system performance. These logs are helpful in assessing the performance of specific Activities, and are useful for troubleshooting. LEI maintains three types of logs:

- Activity Logs – These keep track of individual Activities and provide performance statistics.
- Operation Logs – These keep track of any errors encountered when running an Activity.
- Server Logs – These logs track the operation and performance of LEI servers.

These logs are described in more detail in Chapter 4 of this manual.

## General LEI Terms

Because LEI works with several database products, directories, ERP systems, and transaction processing systems, it's important to understand the vocabulary used in LEI and in other data sources.

| *Term* | *Description* |
|---|---|
| Column | Equivalent to an LEI field, a set of similar data such as phone numbers or customer names, usually displayed in column format. A column contains the fields having the same position in the data records of the metadata. In a Notes view, fields appear in a column presentation. |
| Database | A collection of metadata and data. A Lotus Notes database is stored as a file with the extension .nsf. Other systems generally abstract the database from the file system. |
| Document | The Notes equivalent of an LEI Record. A Notes document is a database entry that contains information. It is the equivalent of a database row and appears in row format in a view. |
| Field | In LEI, the metadata description for one element of a record. Each record contains one or more fields, each with a data type, data value, and other properties.<br>In Lotus Notes, a specific kind of data represented as an item within a form or document. |
| Form | The standard metadata object in Notes. A form defines the fields and data types in a set of documents in a Notes database. |

*continued*

| *Term* | *Description* |
|---|---|
| Index | A collation of record key values enabling rapid keyed searches. In Lotus Notes, a View contains an Index. In relational databases, an Index is a separate object. |
| Metadata | An object that describes data format and contains data records, such as a database table, a data file, a Notes form, or other similar unit of data. |
| Parameter | Data is passed to a procedure using input parameters. For systems which do not support result sets returned from procedures, data is returned through output parameters. |
| Procedure | A packaged set of operations in an external system. Procedures (also called Stored Procedures or RPCs - Remote Procedure Calls) are executed to produce a result set, modify external data, or perform some other action. Procedures obtain input data through parameters, and return data either through output parameters or returned result sets. |
| Record | An element of data within a metadata object or result set, composed of one or more fields. Examples are a SQL row and a Notes document |
| Result set | The set of data (records) resulting from a selection operation. For example, the data to be transferred after being selected through a Notes select statement or SQL statement would be the result set. |
| Row | In a relational database, the equivalent of a record. |
| Statement | A statement in a system's native language syntax. Statements can produce result sets (selection statements) or invoke modifications, commands, or procedures. Examples include a Notes formula or SQL query. |
| Table | In a relational database, the equivalent of LEI Metadata. A set of data in a relational database, stored in row and column format. |
| View | In Notes, a view collates and displays a list of documents with selected fields.<br><br>The list can be categorized and hierarchical. In SQL databases, a view is a customized presentation of data from one or more tables. |

## Lotus Enterprise Integrator on the AS/400

This section provides information about LEI specific to its use and implementation on the AS/400 platform.

### Lotus Enterprise Integrator features on the AS/400

The features that LEI provides are as follows:

#### LEI Server on the AS/400

On the AS/400, the LEI server is a 64-bit application which will execute as an addin server task to a native 64-bit Domino server on the AS/400. Access to DB2 databases, Notes databases and File System objects is allowed via the LEI server running on the AS/400. Direct (native API) Connections to databases such as Oracle and Sybase are not supported on the AS/400, nor is the ODBC Connection allowed. Access to non-DB2 data sources can be achieved by using IBM's DataJoiner product on an NT IPCS server, outboard NT, AIX platform to reroute DRDA requests from AS/400 to the target data source. Connections to Sybase, Oracle and other ODBC accessible databases can, however, be registered in an administration database for LEI servers configured to run on non-AS/400 platforms in your LEI cluster.

#### Development Client on the AS/400

On the AS/400, there is no distinct concept of an LEI Development Client. To browse fields for field mapping or metadata selection, you would install the LEI Development Client on your workstation in accordance with the operating system supported there. The same would be true for developing and testing Lotus Enterprise Integrator LotusScripts.

#### LEI Server Running Window on the AS/400

Unlike the other LEI platforms, the AS/400 does not have an LEI server window or console. The running state of LEI can be viewed from the LEI administration database or from the Domino console on the AS/400 (via show tasks). The LEI server is an addin task to the AS/400 Domino server and so it is not launched in its own interactive process or window. Commands (such as Shutdown Server, Stop Server, Close Activity, and Kill Activity) that can be issued in the server window on other server platforms are available only from the Action menu of the LEI Administrator on AS/400. The Reconfigure option is now automatic. When the LEI Server configuration document is changed, the corresponding server will reconfigure during its next polling cycle or when the server is restarted, whichever happens first.

## Languages Support on the AS/400

The LEI is considered an administrative tool and is therefore currently available in English only. All user interfaces (AS/400 CL commands, help text, messages, and the LEI administration databases) are in English only. The English language (2924) does not have to be installed on your AS/400.

### Code Page Translation Support on the AS/400

Data translation is automatic (not optional) for LEI on the AS/400 because the AS/400 is an EBCDIC platform and thus is incompatible with Domino's LMBCS code page. LEI supports data translation in a variety of language code pages. See Appendix C for a list of supported code pages.

LEI Activities launched under the LEI server will default to the job CCSID of the Domino server. See AS/400 Domino server documentation for how to set the default job CCSID (the locale information specified in the QNOTES user profile) for Domino Server addins.

All LEI Activity functions are supported for the AS/400, including the Java Activity and Java classes.

The following LEI Connectors and MetaConnectors are supported for AS/400:

- Connectors:  DB2, Notes, File, Text
- MetaConnectors:  Collapse/Expand, Connection Broker, Meter, Order, Trace

## Term Comparison for the AS/400

The following table compares key terms that you will find in this manual.

| *Term* | *AS/400 Relational DB* | *SQL* | *Notes* |
|---|---|---|---|
| Database | Any RDB registered via WRKRDBDIRE, including the local DB2/400 database | Database | Database (.nsf file) |
| Metadata | File | Table | Form |
| Record | Record | Row | Document |
| Field | Field | Column | Field |
| Statement | | Query | Selection formula |

# Chapter 2
# Starting and Running the LEI Server

This chapter provides information and instructions for starting and running the LEI server on the supported operating systems.

## Starting the LEI Server

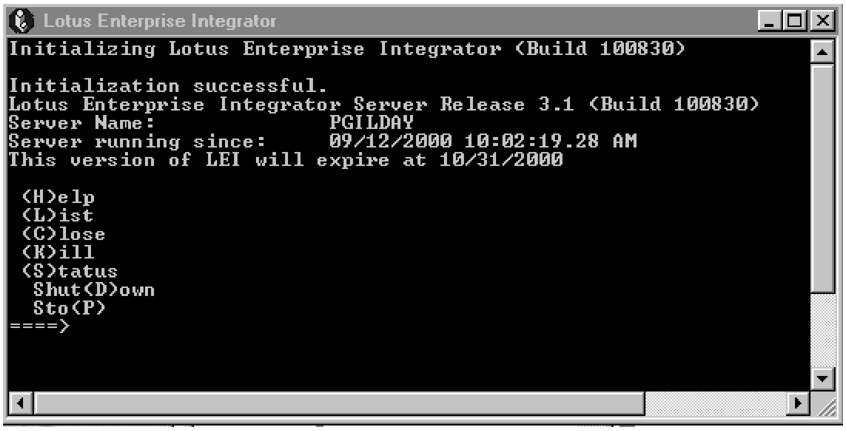
The LEI server must be running in order to execute LEI Activities.

Start the LEI server by executing the LEI program at the LEI server machine. The LEI program name varies by platform.

| *Platform* | *Name* |
|---|---|
| Win32 | nlei.exe |
| All UNIX | lei |

On Windows, you can also start the LEI server by selecting it from correct program group folder using the Start menu.

The LEI console appears, displaying the LEI server commands.



```
Initializing Lotus Enterprise Integrator (Build 100830)

Initialization successful.
Lotus Enterprise Integrator Server Release 3.1 (Build 100830)
Server Name:                PGILDAY
Server running since:       09/12/2000 10:02:19.28 AM
This version of LEI will expire at 10/31/2000

 (H)elp
 (L)ist
 (C)lose
 (K)ill
 (S)tatus
  Shut(D)own
  Sto(P)
====>
```

When started, the LEI server connects to the LEI Administrator database and immediately runs any overdue Activities. It continues to poll the Administrator database at the interval defined in the server configuration document and to run scheduled Activities until shut down.

**Note** LEI is not supported on partitioned servers.

## Executing LEI Server Commands From the Console

The following LEI server commands can be executed from the server console window:

The LEI server commands are described below.

| *Command* | *Description* |
|---|---|
| Help | Displays a brief description of the LEI server commands. |
| List | Displays a list of Activities that are currently being executed by this LEI server. The number associated with each Activity name must be used with the Close and Kill commands. |
| Close | Close is the preferred method (over the Kill command) of prematurely terminating an Activity. One or more numbers (separated by spaces) may be supplied, each representing an Activity to close. These numbers are obtained with the List command. A request for termination is sent to the relevant Activity process, which acts following the current database operation.<br>When an Activity is closed, LEI does not consider it an error. LEI will not mark the Activity as failed, will not send mail notification, and will not execute dependent Activities.<br>To immediately close an Activity process which is not responding to a close request, use the Kill command. |
| Kill | Immediately terminates an Activity process. One or more numbers (separated by spaces) may be supplied, each representing an Activity to kill. These numbers are obtained with the List command. Regardless of the current Activity state, it will immediately be terminated. The preferred method of stopping an Activity is to use the Close command, which provides an opportunity for the Activity to clean up its database connections and log the termination.<br>Killing a RealTime Notes Activity should not be done, as it will result in the monitoring process for that Activity remaining enabled until all Notes and LEI processes are stopped. |
| Status | Displays basic configuration information about this LEI server, as well server time and state. |

*continued*

| *Command* | *Description* |
|---|---|
| Shutdown | Terminates the LEI server. This command is preferred over the Stop command for terminating the LEI server. It prevents the LEI server from starting any new Activities and waits for all currently running Activities to shut down. When all Activities have shut down, it stops the LEI server program. For a rapid termination of the server, use the Stop command. |
| Stop | Quickly terminates the LEI server. It prevents the LEI server from starting any new Activities and sends a Close command to all currently running Activities. After a period of time (specified in the Server Configuration document's Activity Shutdown Requested Timeout field), any Activities which are still running are Killed (see the Kill command) and the LEI server program stops. To shutdown of the server, use the Shutdown command. |

The Shutdown, Stop, Close, and Kill commands are also available under LEI Server Administration and Activity Administration in the Actions menu of the LEI Administrator. For example, to close a running Activity, select the Activity from an Administrator view and select the Activity Administration/Close Activity command from the Notes Actions menu in the LEI Administrator. The next server polling cycle will pick up the request to close the Activity and display a message indicating that the Activity is being closed.

When issuing commands in the LEI console, the command text is limited to 255 characters.

## Executing LEI Server Commands from the Administrator

The following LEI server commands can also be executed from within the LEI Administrator:

- Close Activity
- Kill Activity
- Shutdown Server
- Stop Server

To execute a server command from the LEI Administrator, select Actions – LEI Server Administration (or Actions – or Activity Administration), and then select the desired command from the submenu. If an inappropriate document is selected when the action is attempted, an error message is displayed.

If you receive an error message, see Appendix D, "Error Messages and Troubleshooting".

## Running LEI as a Domino Addin Task

You can run LEI as a Domino server addin task. This option is available when you install LEI. This option automatically starts LEI when the Domino server starts, and shuts down LEI as the Domino server is stopping. If you did not select this option during install, you can enable it manually by editing the Domino server's notes.ini file and adding the task "LEI" to the server task list.

LEI can also be manually controlled as an addin task.

To start LEI as an addin manually, enter the following command from the Domino server console:

```
LOAD lei addin
```

To manually shut down LEI when running as an addin task, enter the following command from the Domino server console.

```
TELL lei QUIT
```

When running as an addin task, the LEI console display is suppressed. To send LEI console commands, submit the following command to the Domino server console.

```
"TELL lei <command>"
```

For example, the following command will have the same effect as entering "c 1" at the standard LEI server console:

```
TELL lei c 1
```

## Considerations When Running LEI on UNIX

The following shell script is provided with LEI installations on UNIX platforms to clean up LEI resources following an abnormal termination of LEI.

```
leiclean
```

Use the leiclean shell script with caution. It kills all LEI processes owned by the UNIX user who executes it. It also frees all shared memory and semaphores currently in use by that user. The removal of shared memory and semaphores could affect other programs that are running under the same user ID, including the Domino server and any other Notes API programs.

If you execute LEI and the Domino server under the same user ID (as you must to use RealTime), an abnormal termination by either process may halt the other. This is a Notes API limitation.

In addition to leiclean, there is a Notes shell script that can be used to clean up Notes/Domino processes after an abnormal termination. To use this, change your directory to the Notes data directory, then execute the following:

```
nsd -kill
```

**Note** The shell script nsd -kill is only available for Domino 5 and up. If you are using a previous version of Domino, you must kill each PID separately.

## Starting and Running the LEI Server on AS/400

There are two ways of starting the LEI server after it is installed – automatically or manually.

### Starting the LEI Server Automatically – as a ServerTask Addin using Notes.ini

If you entered *YES (the default) for the Autostart LEI parameter on the ADDLEISVR command, then the notes.ini was updated to include LEI in the server task list. Therefore, whenever the Domino server is started, LEI is also started. If you did the ADDLEISVR and have not restarted the Domino server, you can use the second option and start LEI from the Domino console.

### Starting the LEI Server Manually – from the Domino Console

On the AS/400, you can enter commands to the Domino server using the Work Domino Console (WRKDOMCSL) command. For example, to work with the console for a Domino server called LEISERVER1, you would enter the following at the AS/400 command line:

```
WRKDOMCSL LEISERVER1
```

From the console, you can enter Notes commands such as the following to start the LEI server:

```
LOAD LEI
```

**Note** With either method, if LEI Administrator databases are located on a different Domino server, that server must be active before LEI will start.

**Note** If you originally configured the LEI server with Autostart LEI as *NO, and now want to automatically start it as Domino server task, you can edit the notes.ini file and add LEI to the server task list.

**Note** There is no action pulldown possible from the LEI Administration database itself to start the LEI server.

## Stopping the LEI Server on AS/400

There are three ways to stop the LEI server:

### From the Domino Console

From the Domino console, you can end the LEI server by issuing the following command:

```
tell lei quit
```

If you have LEI Activities currently running, they will end as soon as they are notified of the quit request. The Activities will not run to completion.

### From the LEI Administrator Database

You have two options from the LEI Administrator Database action pulldowns via the *RUNNING view:

- Shutdown – end LEI in an orderly fashion; Activities run to completion.
- Stop – end LEI immediately; Activities do not run to completion.

### End the Domino Server

By issuing an ENDDOMSVR server-name command, the LEI server will also end in a controlled manner.

If you have LEI Activities currently running, they will end as soon as they are notified of the server stopping. The Activities will not run to completion.

If you end the Domino server using the *IMMED option, the jobs end immediately and any LEI Activities currently active are terminated. This can cause unexpected results, such as incomplete transfers or lengthy rollbacks. You should only use the *IMMED option if all other methods, including *CNTRLD, fail.

## Determining if LEI Server is Running on AS/400

There are three ways to determine if your AS/400 LEI server is running:

### Show Task from Domino Console

- WRKDOMCSL domino-server-name
- show tasks
- F5 – REFRESH

The following tasks should be included in the list. There may be additional LEI tasks listed if Activities are running. These will have the name of the Activity in the description.

| | |
|---|---|
| • LEI Server | Administrator is running |
| • LEI Server | Server is running |
| • LEI Server | Activity Transfer Daily Issues is running |

## WRKACTJOB on AS/400

When you enter the command WRKACTJOB, you will see the following LEI jobs under the subsystem for the Domino server. If Activities are running, there will be an LEIENGAI job running for each running Activity.

```
DOMINOxx
    . . .
    . . .
    LEICSMAI
    LEI
```

## From LEI Administrator Database

Select the *RUNNING view and look at the active server to see the list.

# Chapter 3
# LEI Development Client

This chapter provides information about the LEI Development Client.

## Overview of the LEI Development Client

The LEI Development Client provides a subset of the functionality provided by the server component, and is installed as an alternative to the Server. It must be installed on a machine with a Notes Client Release 4.6 or greater and is only supported for Windows. It allows you to use the database browsing capabilities found within the LEI Administrator, metadata selection, and field mapping from a local Notes client. This gives you the ability to develop Activity and Connection documents in an Administrator database. A development client also provides the needed support files to enable LEI LotusScript development. Scripts can be verified and debugged on the local machine running the development client.

**Note** Appropriate database software must also be installed for the data sources that you plan to access (Oracle SQL *NET, DB2 CAE, ODBC, and so on).

**Note** A Lotus Domino server is not required in order to run the LEI development client.

## Installing the Lotus Enterprise Integrator Development Client

Run the LEI Setup program on the computer whose Notes Client will host the Development Client. Select the “Install LEI Development Client” option. For more information, see the *LEI Domino Connectivity and Installation Guide*.

You must use either a Notes ID without a password or the option “Share password with Notes add-ins” must be selected.

Installation of the LEI Development Client is not supported for UNIX hosts.

**Note** The LEI Development Client does not need to be installed in addition to an LEI Server. The LEI Server includes all components necessary for creating Connections, Activities and scripts using the LEI LotusScript Extensions.

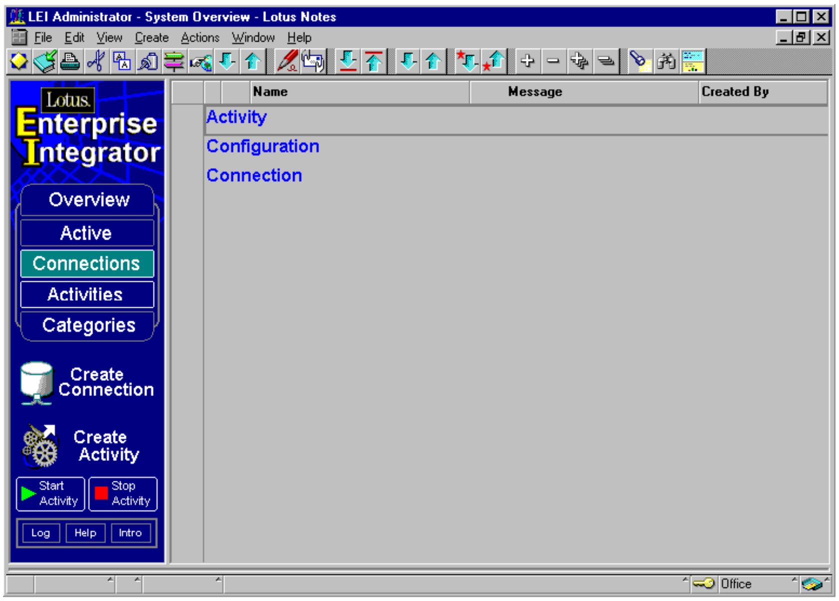
# Chapter 4
# LEI Administrator

This chapter provides information about the LEI Administrator.

## Introduction to the LEI Administrator

The LEI Administrator is a Notes application. The Administrator database contains all of the configuration, Activity, Connection, and scheduling elements that LEI servers use to collect, transfer, and write data. The Administrator database is also known as the Control Store.

The LEI Administrator interface is shown in the figure below. The LEI Navigator is shown on the left. The Navigator enables you to select different views, which are described in the next section.



| Name | Message | Created By |
|---|---|---|
| Activity | | |
| Configuration | | |
| Connection | | |

## The LEI Administrator Database

A single Administrator database can hold information used by several LEI servers, but each LEI server can be governed by only one Administrator database. A group of LEI servers governed by one Administrator is an LEI cluster.

The Administrator interacts with a Log database that documents the processing and outcome of Activities and server operations. This is described later in this chapter.

**Note** Do not use Notes Replication on the LEI Administrator database. LEI server operations such as scheduling and status checking will be seriously disrupted.

## LEI Navigator

The LEI Navigator provides buttons for creating Connections and Activities and view selections for looking at different aspects of your LEI environment. The menu bar provides access to the actions and views also.

### Overview

The Overview view lists all documents: Activity, Configuration, and Connection documents, along with each document author's name. It also displays the status of an Activity or server.

### Active

Click Active to view currently running Activities and servers.

### Connections

Click the Connections button to view, by type, all of the Connection and MetaConnection documents that have been created on the system.

### Activities

Click the Activities button to view, by type, all of the Activities that have been created on the system. Each column of this view is sortable. Click the column header to sort. For example, to see all Activities by name, instead of by type, click the Name column.

## Categories

The LEI Categories view lets you view all of your Connections and Activities according to user-defined categories. Each Connection and Activity document has a field at the bottom for entering a category.

## Create Connection

Click Create Connection to create a new Connection document. A dialog box appears after you select this option, allowing you to select the type of Connection or MetaConnection you want to create.

## Create Activity

Click Create Activity to create a new Activity document. A dialog box appears after you select this option, allowing you to select the type of Activity you want to create.

## Start Activity

Click Start Activity to place the selected Activity in the queue, which will cause it to run the next time the LEI server polls the LEI Administrator. You must be in an Activities view and have selected the Activity before selecting this option.

## Stop Activity

Click Stop Activity to stop a currently running Activity. You must be in an Activity view and have selected the Activity before selecting this option.

## Log

Click Log to jump to the LEI Log database (leilog.nsf). The log lists all LEI log entries categorized by either LEI server or type of log entry (Activity, Operation, or Server).

## Help

Click Help to open the online version of the LEI User Guide.

## Intro

Click the Intro to access an online description of the LEI Navigator. This description includes a guide to the screen icons that are used in many of the views.

## LEI Administrator Menu Commands

In addition to the standard Notes commands found in the menus, the LEI Administrator menus provide specific commands for working with LEI.

## View Menu Commands

The Administrator View menu includes all of the commands and views available through the Navigator as well as the following:

**Activities**
The Activities view gives you the option of viewing by Name or by Type. In the By Type view, each column is sortable by either Name or by Type by clicking the column header.

**Configuration**
The Configuration view shows the Administrator Configuration and Server Configuration documents.

**Connections**
The Connections view gives you the option of viewing by Name or by Type. In the By Type view, each column is sortable by either Name or Type by clicking the column header.

## Create Menu Commands

The Administrator Create menu includes the commands described below.

**Activity**
The Activity submenu provides a list of all LEI Activities that can be created. Select the type of Activity you want to create. To create an Activity that is not included in the submenu, click Other to get the complete list. For more information about creating Activities, refer to the chapter in this manual about the type of Activity you want to create.

**Connection**
The Connection submenu provides a list of all Connections that can be created. Select the type of Connection you want to create. For more information about creating Connections, refer to Chapter 5, "Introduction to Connectors."

**MetaConnection**
The MetaConnector submenu provides a list of all MetaConnections that can be created. Select the type of MetaConnection you want to create. For more information about creating a MetaConnector, refer to Chapter 16, "MetaConnectors."

## Actions Menu Commands

The Administrator Actions menu includes the commands described below.

### Activity Administration

The Activity Administration submenu provides commands related to Activities.

**Clear Browsing Cache**
Caching within the LEI Administrator provides a way of retaining previously defined metadata mapping selections. What caching means in the context of the browser (the Select Metadata, Map Fields and Map Timestamp action buttons) is that the lists of tables/forms and fields are retained so that a new Connect and Fetch of the metadata is not required each time. The caching is accomplished by using a special type of document in Notes called a Profile document. Profile documents are hidden, and they persist across Notes sessions.

The Profile document for caching is linked to the Activity by its name. It is possible to have one Profile document for each Activity. The Profile form used in the LEI Administrator is called "Catalog Profile." Below is an outline of the correct behavior for caching.

| | |
|---|---|
| New Activity | Browser should Connect and Fetch information as required by the user, caching the metadata for future use. |
| Existing Activity | Browser will populate the lists with metadata taken from the cache. If a new table/form is selected (other than what the cache contains), a Connect and a Fetch of the new list of fields will result. If a Connection is changed, the cache is similarly rebuilt (e.g., a Connect and Fetch occur) starting with the table/form list. |

When an Activity is deleted, any existing Profile document is also destroyed. An Agent (Clear Browsing Cache) exists to destroy a Profile document from the Activity document level on demand.

A Refresh button is available from all browsing dialogs which include metadata lists. This connects to the data source and fetches a new copy of the metadata. This button is not available when no cache currently exists (e.g., in the case of a new document or if different metadata is chosen from what currently exists in the cache) or immediately after a refresh has occurred. This prevents needless Connections and fetches.

**ClearLock**
The ClearLock Agent clears an Activity document's Lock field so that the Activity can be run again.

The purpose of the agent is to provide a manual way to clear the lock in case an LEI server goes down while running the Activity and you want to run the Activity before the automated server cleanup occurs.

Select the documents from a view before running this agent. The agent ignores non-activity documents.

**Close Activity**
This command closes a selected Activity that is currently running. Close is the preferred method (over the Kill command) of prematurely terminating an Activity.

When an Activity is closed, LEI does not consider it an error. LEI will not mark the Activity as failed, and does not send mail notification on error. The Activity is not a completed Activity and dependent Activities are not executed.

To immediately close an Activity process, use the Kill command.

**Kill Activity**
This command immediately terminates an Activity process.

**Note** The preferred method of stopping an Activity is the Close command, which provides an opportunity for the Activity to clean up its database Connections and log the termination. Using the Kill command may leave the system in an unstable state due to improper termination of the Activity, so only use it when absolutely necessary. LEI considers killing an Activity an error.

**Set Designated Server**
This Agent is available from the Actions menu. Use it when a Server Configuration document has been unintentionally destroyed. Since LEI identifies servers through an ID rather than by name, all Activities using the server field will be synchronized with the new server ID. The Agent acts on selected Activities.

### Enable Selected Activities
This command enables scheduling for Activities whose scheduling is currently disabled.

## LEI Server Administration

The LEI Server Administration submenu provides commands related to the operation of the LEI server. These commands are described below.

### Shutdown Server

This command terminates the LEI server.

Shutdown Server is the preferred method (over the Stop Server command) of prematurely terminating a server. It prevents the LEI server from starting any new Activities and waits for all currently running Activities to shut down. When all Activities have shut down, it stops the LEI server program.

To immediately terminate the LEI server, use the Stop Server command.

### Stop Server

This command immediately terminates the LEI server.

It prevents the LEI server from starting any new Activities and sends a Close command to all currently running Activities. After a period of time (specified in the Server Configuration document's Activity Request Shutdown Timeout field in the Administrator database), any Activities which are still running are Killed (see the Kill command) and the LEI server program stops.

**Note** The preferred method of terminating the LEI server is the Shutdown Server command,

## RunASAP

RunASAP causes the selected Activities to be executed by the LEI Server the next time the server polls the LEI Administrator database.

There are two ways that you can run this agent:

- **From a View**: Select one or more Activity documents in a view and choose RunASAP from the Actions menu to launch the selected Activities.
- **From within an Activity Document**: Click the RunASAP button in the Action toolbar (or choose RunASAP from the Action menu) to have the open Activity execute. The Run ASAP button is available once a document has been saved. The button is not available in Edit mode.

Unless you built a schedule that excludes the Activity from executing at the time it's created (such as eliminating the current day from the Days of the Week), an Activity will generally be queued to run for the first time as soon as it is saved.

## How to Create, Edit, or Delete LEI Documents

LEI Connection, MetaConnection, and Activity documents are created through the LEI Administrator. Administrator and Server Configuration documents are created when LEI servers are installed.

### Creating New LEI Documents

To create a LEI document:

1. Start Lotus Notes.
2. Open the LEI Administrator database.
3. Choose Create from the Notes menu (or choose Create Connection – Create Activity from the Navigator. If you use the Navigator, go to step 5).
4. Choose the document type that you want.
5. Fill in the document form according to the type of document you are creating. Refer to the appropriate chapters of this manual for information about creating specific types of Connections, MetaConnections, and Activities.

   **Tip** Dark blue text means pop-up Help is available. For a brief Help note, click the dark blue text and keep the left mouse button pressed.

   When completing the fields, follow these guidelines:

   - Make your entries within the edit field brackets.
   - Some entries are made by selecting from a key list. The labels of these lists are followed by the standard Notes entry helper button. Click the arrow to the right of the field to see the key list and click the selection that you want to make.
   - Small blank buttons on the form give you access to additional options when clicked.
   - Red field brackets indicate encryption-enabled passwords.
   - All documents end with optional Category and Comments fields that can help you manage your documents:

     **Category –** Entering a category name will let you group a set of documents Activities by Category view. If the category you enter doesn't exist, it will automatically be created. You can enter multiple categories, separated by commas.

     **Comments –** Enter text to describe the document. This is a rich text field so that you can place anything here, including links to other documents.

6. Save the completed form by choosing Save in the File menu or by clicking the Save & Exit action button at the top of the form.

**Note** You can use Notes Reader and Author list privileges to add additional security to Activity and Connection documents. To add or update Authors of the document, click the Author Privileges action button for a list of three options for managing Author access. Select an option and click OK to use it. Document authors are people who will have authorization to edit the document once it's created.

| | |
|---|---|
| Add name[s] to document editor list | Select this option to add a name(s) from the Notes Public Name and Address Book to the list of editors for this document. |
| Remove a name from document editor list | Choose this option to delete a name from the list of document editors. |
| View document editor list | Choose this option to view the list of current editors for the document. |

## Editing Existing Documents

Any document (Activity, Connection, MetaConnection, and Configuration) can be edited.

To edit an existing document:

1. Open the Administrator database.
2. Choose a view from the LEI Navigator (or from the View menu):
3. When the document appears, double-click the name of the document in the view.
4. Enter the Edit mode using one of the following methods:
   - Press Ctrl-E
   - Choose Edit Document from the Action menu
   - Choose the Edit Document button at the top of the form
   - Double-click anywhere in the document
5. Make the desired changes to the document.
6. Save the document by choosing File – Save or by pressing the Esc key and choosing "Yes".

### Deleting Documents

When you no longer need a document, you can delete it. Be careful in considering deletion of Configuration documents; they should be deleted only by the Setup program. You need appropriate access rights to delete LEI documents.

**Note** When deleting Connection documents, you must know whether or not they are used by other documents. Deleting a document whose name remains in another document will cause an error message when the Activity is run or when you try to open a dependent document.

To delete a document:

1. Open the Administrator database.
2. Choose a view from the LEI Navigator (or from the View menu):
3. Select the document you want to delete and press the Delete key.

   The document will be deleted when you close or refresh the view.

## LEI Administrator Action Buttons

LEI Administrator documents may include the following action buttons:

- Author Privileges
- Edit Document
- Map Fields
- Map Timestamp
- Run ASAP
- Save & Exit
- Select Metadata

**Note** If you receive the error "Error loading USE or USELSX module: *lsxlei" when you click one of the browsing related buttons (Select Metadata, Map Fields, or Map Timestamp), you either don't have the LEI client or server installed on the machine that you are trying to run Browsing from, or the LEI directory is not in your path.

## Understanding Browsing

Prior to LEI Release 3.1, when the list of tables to select from was too large, an overflow error message would appear. In this event, you would be required to manually enter the exact table name into the field.

Now, when the list is longer than the output window can accommodate, a message will appear. A Next and Previous option in the display window provide additional viewing.

To see this, do the following:

1. In LEI, open an existing Activity such as Direct Transfer.
2. Choose the Select Metadata button.

   You can optionally select the "Filter by Owner Name" option to limit the list of displayed items based on a specific table owner name.
3. As prompted, select the Source or Target menu option. A list or message will appear:
   - If the list of items retrieved can be fully displayed, you can simply select from the resultant list.
   - If the list of items retrieved is greater than what the system can display on one page, the Next and Previous options will appear at the bottom of the output window. Use these options to toggle between the two pages of output.
   - If the list of items is larger than what the system can display in two output pages, you'll receive a message stating that fact and supplying options to continue or to modify the table search by table owner.

**Note** The list does not buffer a set number of entries. Rather, it retrieves and lists based on size and bytes. For example, long entry names yield view output with fewer items than do shorter item names. Item names consist of owner name followed by table name.

**Note** The Select Metadata button is optional. You can manually enter the table name.

## Author Privileges

The Author Privileges button, shown below, allows you to change author privileges for the document currently being edited.

### Using the Author Privileges Action Button

To change Author Privileges for the Activity:

1. Click the Author Privileges action button. The Author Privileges dialog box appears.

   **Note** You can use Notes Reader and Author list privileges to add additional security to Activity and Connection documents. To add or update Authors of the document, click the Author Privileges action button for a list of three options for managing Author access. Select an option and click OK to use it. Document authors are people who will have authorization to edit the document once it's created.

2. Choose the action you want from the options available in this dialog box. These actions include:

| | |
|---|---|
| Add name[s] to document editor list | Select this option to add a name(s) from the Notes Public Name and Address Book to the list of editors for this document. |
| Remove a name from document editor list | Choose this option to delete a name from the list of document editors. |
| View document editor list | Choose this option to view the list of current editors for the document. |

## Run ASAP!

The Run ASAP! button causes the Activity to be run the next time the LEI server polls the LEI Administrator database.

### Using the Run ASAP! Action Button

Click the Run ASAP! button when you want to run an Activity as soon as possible. The LEI runs the Activity the next time the LEI server polls the LEI Administrator. Run ASAP is available only from an Activity document after the document has been saved.

## Select Metadata

The Select Metadata button, shown below, lets you display available metadata objects from which you can pick to use in building Activities.

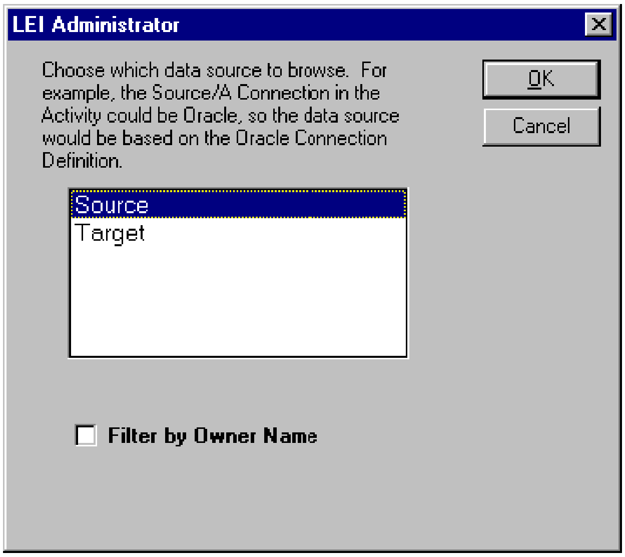
If the Activity has two Connections, run Select Metadata for each.

### Using the Select Metadata Action Button

Select Metadata is available when an Activity document is in Edit mode.

To use the Select Metadata action button:

1. Click Select Metadata.
2. Select the metadata type you want from the dialog box. The metadata type option depends on the Activity type.
   - Archive and Direct Transfer Activities: Click Source or Target, depending on which Connection you want to browse.
   - Replication: Click Connection A or Connection B.
   - Polling: No option. Polling uses a single Connection.
   - RealTime: Click Notes or External Data Source.


LEI Administrator

Choose which data source to browse. For example, the Source/A Connection in the Activity could be Oracle, so the data source would be based on the Oracle Connection Definition.

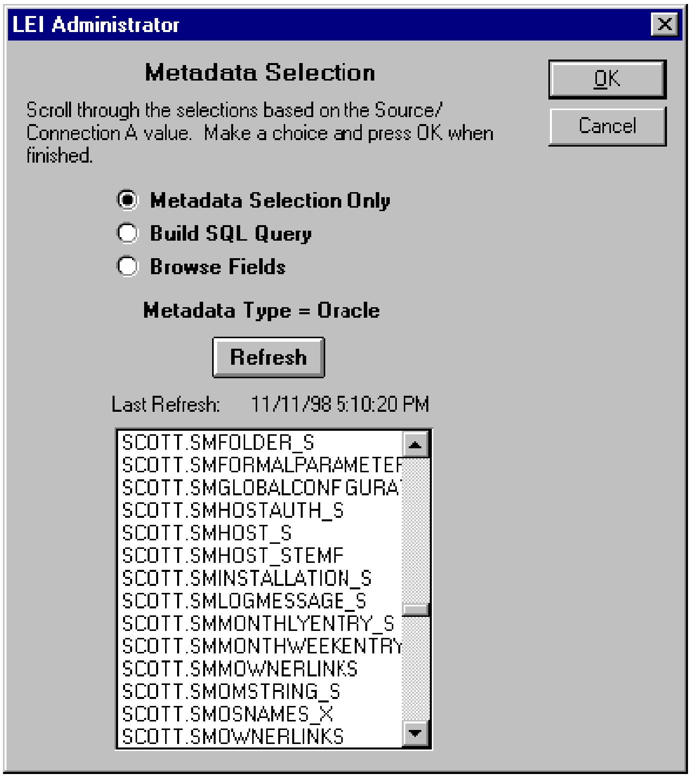
**Note** For information about the "Filter by Owner Name" option, see the "Filter by Owner Name" section, below.

3. Click OK. The Metadata Selection dialog box appears.

   If you are working with the Source, you will see the field selections. If you are working with the Target, the metadata is entered into the form and the action is completed.

4. Choose Select Metadata Only (which is the default), Build SQL Query (if your data source supports SQL) or Browse Fields.

   If you select Build SQL Query or Browse Fields, you'll see the Field Selection dialog box. Otherwise the metadata is entered into the form and the action is completed.



Metadata Selection

Scroll through the selections based on the Source/Connection A value. Make a choice and press OK when finished.

[✓ Metadata Selection Only] Build SQL Query Browse Fields

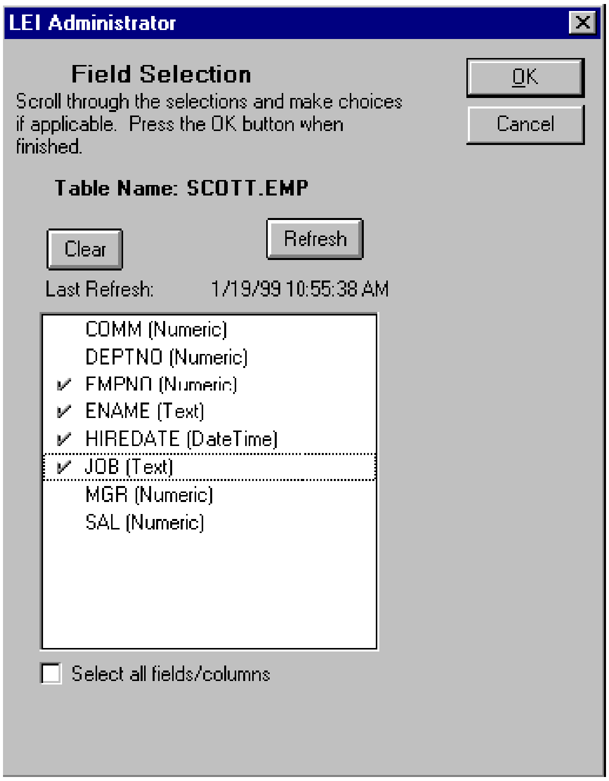
Metadata Type = Oracle

Last Refresh: 11/11/98 5:10:20 PM

```
SCOTT.SMFOLDER_S
SCOTT.SMFORMALPARAMETEF
SCOTT.SMGLOBALCONFIGURA
SCOTT.SMHOSTAUTH_S
SCOTT.SMHOST_S
SCOTT.SMHOST_STEMF
SCOTT.SMINSTALLATION_S
SCOTT.SMLOGMESSAGE_S
SCOTT.SMMONTHLYENTRY_S
SCOTT.SMMONTHWEEKENTRY
SCOTT.SMMOWNERLINKS
SCOTT.SMOMSTRING_S
SCOTT.SMOSNAMES_X
SCOTT.SMOWNERLINKS
```

- Metadata Selection Only – Lets you choose from a metadata list based on the selected Connection.
- Build SQL Query – If your data source supports SQL, lets you build a SELECT query by selecting fields.
- Browse Fields – Will not replace any existing command statement. (Command statements exist in Direct Transfer and Polling Activities).

The Field Selection dialog is shown below. The field selection options depend on the kind of Activity.



Scroll through the selections and make choices if applicable. Press the OK button when finished.

Table Name: SCOTT.EMP

Last Refresh: 1/19/99 10:55:38 AM

- COMM (Numeric)
- DEPTNO (Numeric)
- ✓ EMPNO (Numeric)
- ✓ ENAME (Text)
- ✓ HIREDATE (DateTime)
- ✓ JOB (Text)
- MGR (Numeric)
- SAL (Numeric)

☐ Select all fields/columns

- Direct Transfer and Polling: If you picked Build SQL Query, choose the fields that you want to appear in the command statement. These will be the fields that LEI will select for transfer or evaluate when polling. Check Select All if you want to choose all the fields in the selected metadata.

  **Note** LEI does not build a select statement using formula language through Select Metadata for a Notes Source.

- For RealTime and Replication Activities, choose the key field or fields.

**5.** Click OK to complete the action.

**6.** Save the Activity. You must save the document in order to save your metadata and field selections.

### Filter by Owner Name

This option allows users to view only those tables owned by a certain owner. It only applies to those data sources that support the concept of owner names. This excludes Domino/Notes, Text and File System Connectors and ERP Connectors.

If this option is checked, the next dialog will contain a list of owners found based on the information provided by the Connection selected. After an owner is chosen, the list of tables owned is displayed.

Owner names are cached in the same way that table and field lists are cached. The same Refresh button and last refresh Datetime appear. Additionally, the owner name filtering checkbox displays the correct setting.

If a Connection previously had Owner Name filtering turned on, it may be turned off and the entire list of tables for that Connection will be displayed. This means that a Connection to the data source is made, new table data is gathered, and the cache is rewritten.

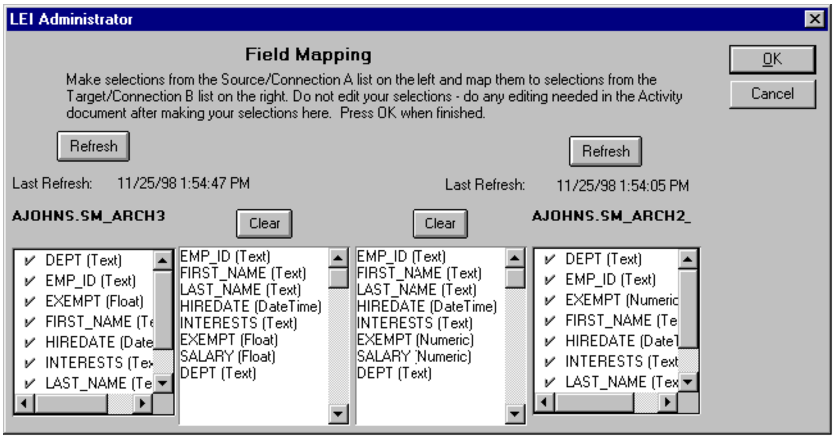
## Map Fields

The Map Fields action button, shown below, displays fields in the metadata. By selecting a list of fields from both metadata objects, you can build a field list that the Activity can use to map fields in moving data.

When you map fields with some Connectors, the result set can be limited by providing an owner name. If more than one metadata entry has the same name and no table qualifier is provided, Map Fields may return fields from all the metadata entries, rather than only the intended one, resulting in duplicate or extra fields.

## Using the Map Fields Action Button

1. After specifying both Activity Connections, click the Map Fields button. The button appears when the Map Fields check box is not set to Automatic and the Activity document is in Edit mode.



Make selections from the Source/Connection A list on the left and map them to selections from the Target/Connection B list on the right. Do not edit your selections - do any editing needed in the Activity document after making your selections here. Press OK when finished.

Last Refresh: 11/25/98 1:54:47 PM

Last Refresh: 11/25/98 1:54:05 PM

AJOHNS.SM_ARCH3

✓ DEPT (Text)
✓ EMP_ID (Text)
✓ EXEMPT (Float)
✓ FIRST_NAME (Te
✓ HIREDATE (Date
✓ INTERESTS (Tex
✓ LAST_NAME (Te

EMP_ID (Text)
FIRST_NAME (Text)
LAST_NAME (Text)
HIREDATE (DateTime)
INTERESTS (Text)
EXEMPT (Float)
SALARY (Float)
DEPT (Text)

EMP_ID (Text)
FIRST_NAME (Text)
LAST_NAME (Text)
HIREDATE (DateTime)
INTERESTS (Text)
EXEMPT (Numeric)
SALARY (Numeric)
DEPT (Text)

AJOHNS.SM_ARCH2_

✓ DEPT (Text)
✓ EMP_ID (Text)
✓ EXEMPT (Numeric
✓ FIRST_NAME (Te
✓ HIREDATE (Date
✓ INTERESTS (Text
✓ LAST_NAME (Tex

2. In the lists of fields in source and target (or Connection A and B) metadata, make a one-to-one selection of fields. The first field on the source selected list will map to the first field on the destination selected list, the second to the second, and so on.
3. If you make an error, click the Clear button and start over, or remove the check from the unwanted item in the master list selection. To update the cache, click the corresponding Refresh button in the dialog box. This will cause LEI to read the data from the Connection and update its cached information.
4. When done mapping fields, click OK.
5. Save the Activity document. You must save the document in order to save your field mapping selections.

## Map Timestamp (Replication Activity)

The Map Timestamp action button displays fields in the source and target metadata of a Replication Activity. It is available only when the Replication document is in Edit mode. You can select the field to be used as the Record Timestamp. This is the field containing the modification date of the record, which will be used to determine which record takes precedence in Timestamp replication.

### Using the Map Timestamps Action Button

To use the Map Timestamps action button:

1. After selecting the Activity Connections and Metadata, click the Map Timestamp button.
2. Choose one timestamp field from each list of fields in Connection A and B Metadata. If you make an error, just choose a different field.
3. When done, click OK.
4. Save the Activity document. You must save the document in order to save your timestamp selections.

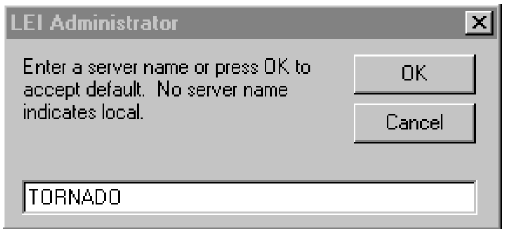
**Note** If you receive the error "Error loading USE or USELSX module: *lsxlei" when you click one of the Browsing related buttons (Select Metadata, Map Fields, Map Timestamp), you either don't have the LEI client or server installed on the machine that you are trying to run the Browsing from, or the LEI directory is not in your path.

## Create Agent (Scripted Activity)

The Create Agent action button, shown below, appears when you create a Scripted Activity.

### Using the Create Agent Action Button

1. Click the Create Agent button. The following screen appears, prompting you to enter a server name for the location of the Scripted Activity.

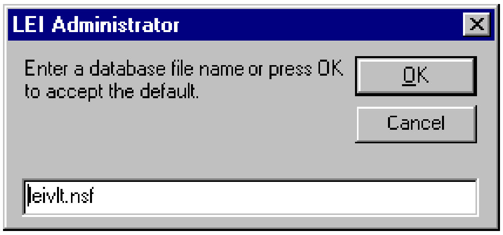


LEI Administrator

Enter a server name or press OK to accept default. No server name indicates local.

TORNADO

2. Enter a server name. To use the default server, click OK.
3. After specifying the server or accepting the default server, the following screen appears, prompting you to specify the database that will contain the Scripted Activity.



LEI Administrator

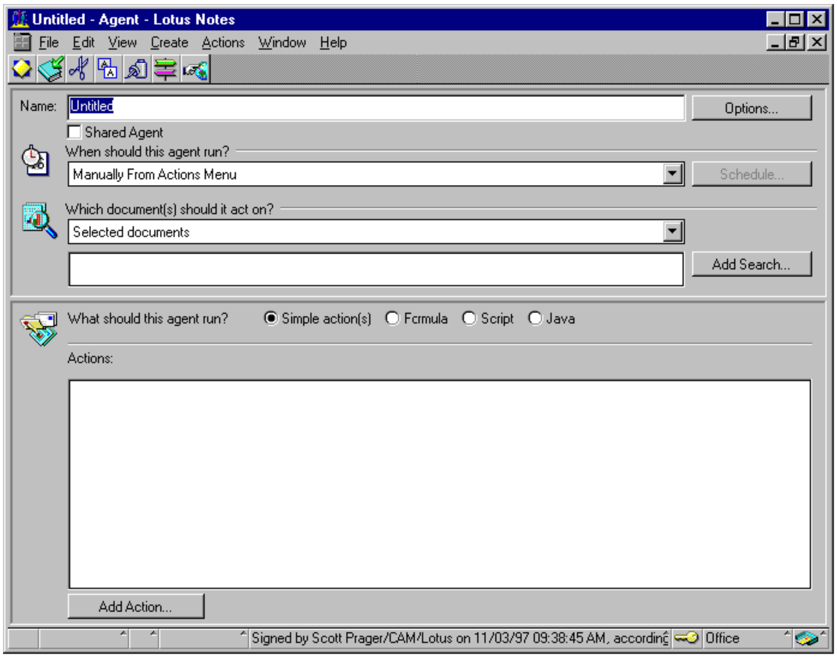
Enter a database file name or press OK to accept the default.

leivlt.nsf

4. Specify the database to contain the Scripted Activity and click OK.

   The LotusScript Agent Development document window appears, as shown below. You use this document to develop your agent.

For more information about Scripted Activities, refer to Chapter 28, "Scripted Activity."


Untitled - Agent - Lotus Notes
File Edit View Create Actions Window Help
Options...
Schedule...
Add Search...
Add Action...

Name: Untitled

☐ Shared Agent

When should this agent run? Manually From Actions Menu

Which document(s) should it act on? Selected documents

What should this agent run? [✓ Simple action(s)] Formula Script Java

Actions:

Signed by Scott Prager/CAM/Lotus on 11/03/97 09:38:45 AM, according Office

## Edit Document

Selecting this action button opens the current document for edit. It also toggles the Save and Exit action button to "on".

## Save and Exit

Selecting this action button saves the changes that you have made to a document.

## LEI Configuration Documents

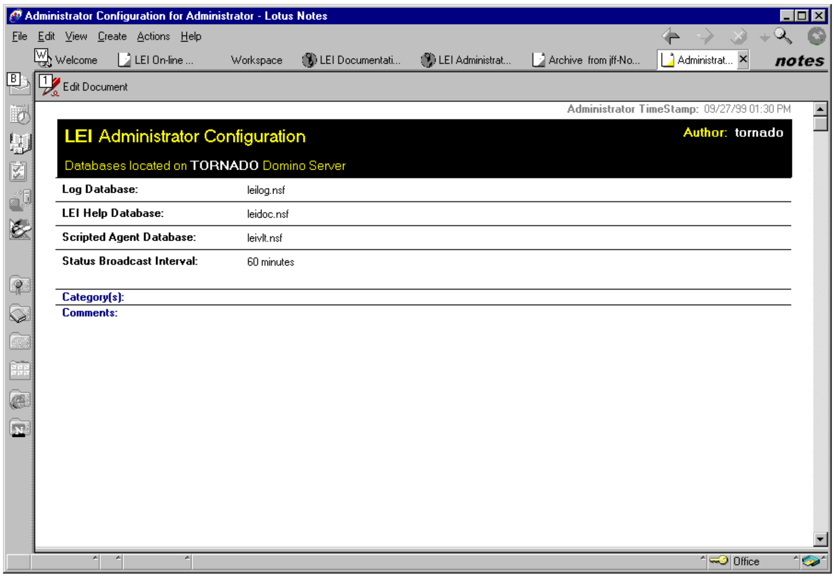
There are three LEI Configuration documents:

- Administrator Configuration document
- Client Configuration document
- Server Configuration document

These documents define parameters for the LEI Administrator databases, LEI Client databases, and the LEI Servers, respectively. These documents are accessible through the Overview hotspot in the LEI Navigator or via the View menu under System Overview.

### Administrator Configuration

Each LEI Administrator database contains a single Administrator Configuration document. The Administrator Configuration document contains configuration information about the Administrator and the LEI servers.



Administrator TimeStamp: 09/27/99 01:30 PM

LEI Administrator Configuration

Author: tornado

Databases located on TORNADO Domino Server

| | |
|---|---|
| Log Database: | leilog.nsf |
| LEI Help Database: | leidoc.nsf |
| Scripted Agent Database: | leivlt.nsf |
| Status Broadcast Interval: | 60 minutes |

Category(s):

Comments:

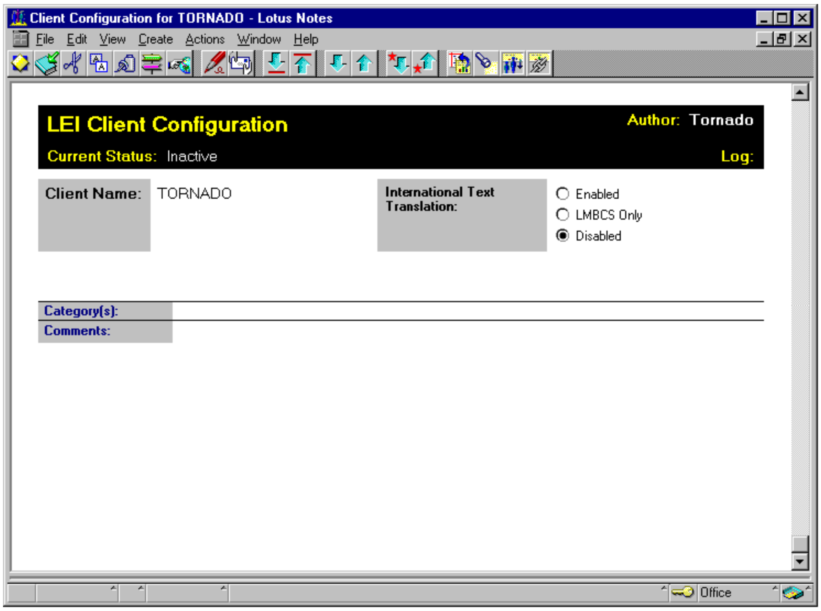
### Administrator Configuration Document

The Administration Configuration document identifies the following information.

| | |
|---|---|
| Log Database | The file path and name of the LEI Log database on the Domino server, relative to the standard data directory. |
| Help Database | The file path and name of the LEI Help database on the Domino server, relative to the standard data directory. |
| Scripted Agent Database | The file path and name of the optional repository for Scripted Agents, relative to the standard data directory. |
| Status Broadcast Interval | The interval, in minutes, at which each running LEI server must notify the Administrator database of its status. Any LEI server that fails to update its status in three of these time periods is assumed to have terminated abnormally.<br>The interval value must be entered in minutes and defaults to a value of 60.<br>The status of the server is reported in the Server Configuration document along with the date and time of the last broadcast received. |
| Category | This is a text field that allows you to enter a category name of your choice. This allows you to group a set of documents by category. If the category you specify doesn't exist, it will automatically be created. You can enter multiple categories per document. |
| Comments | This field holds whatever text you want to put here to describe the document. This is a rich text field (RTF), so you can place anything here, including document links. |

## Client Configuration

The LEI Administrator contains a Client Configuration document for each LEI Client. The document contains configuration information relevant to the LEI Client. An example is shown below.


Client Configuration for TORNADO - Lotus Notes
File Edit View Create Actions Window Help
Office

LEI Client Configuration

Author: Tornado

Current Status: Inactive

Log:

Client Name: TORNADO

International Text Translation: ○ Enabled ○ LMBCS Only [✓ ● Disabled]

Category(s):

Comments:

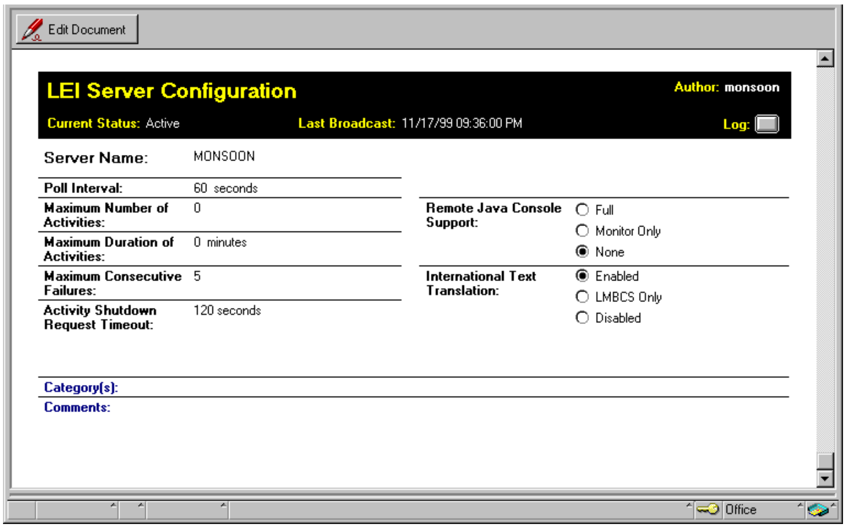
### Client Configuration Document

The Client Configuration document identifies the following information:

| | |
|---|---|
| Current Status | The status of the client. This information cannot be modified. The status is Active or Inactive |
| Client Name | Name you assign to the client at install time. |
| International Text Translation | This field can have one of three settings:  Disabled, LMBCS Only, or Enabled.<br>Disabled – causes the LEI to assume all data is in compatible character sets and transfer the data without translation.<br>Enabled – causes the LEI server to automatically detect the database character set of the source and destination databases and perform translations of the data from source to target. Enabled text data translation causes LEI to perform all necessary character set translation during the transfer of data.<br>LMBCS – Only causes LEI to assume that all non-Notes data is in compatible character sets and only transfers to or from Notes will require translation.<br>International character sets supported by the LEI server are listed in Appendix C, "Character Sets." Support for over 200 different character mappings is provided, including US English, Asian, Far East, Western and Eastern European languages, as well as multi-lingual character sets such as LMBCS (Lotus Multi-byte Character Set) and Unicode. Using the LEI text translation capabilities, for example, an organization with business units in France and China may generate information in French at the source data server, and have it received appropriately in Chinese format by the destination data server located in China.<br>For installations where the Operating System's character set should be used regardless of the database character set, use the LMBCS Only setting. This will force all non-LMBCS character sets to be considered to be in the platform character set. |
| Category | This is a text field that allows you to enter a category name of your choice. This allows you to group Activities by category. If the category you enter doesn't exist, it will automatically be created. You can enter multiple categories per document. |
| Comments | This field holds whatever text you want to put here to describe the document. This is a rich text field (RTF), so you can place anything here including links to documents. |

## Server Configuration

The LEI Administrator contains a Server Configuration document for each LEI server that it controls. The document contains configuration information relevant to the LEI server. An example is shown below.



**LEI Server Configuration**

Author: monsoon

Current Status: Active

Last Broadcast: 11/17/99 09:36:00 PM

Log:

Server Name: MONSOON

Poll Interval: 60 seconds

Maximum Number of Activities: 0

Maximum Duration of Activities: 0 minutes

Maximum Consecutive Failures: 5

Activity Shutdown Request Timeout: 120 seconds

Remote Java Console Support: Full / Monitor Only / [✓ None]

International Text Translation: [✓ Enabled] / LMBCS Only / Disabled

Category(s):

Comments:

## Server Configuration Document

The Server Configuration document identifies the following information:

| | |
|---|---|
| Current Status | The status of the server. This information cannot be modified directly. The status is Active or Inactive. |
| Last Broadcast | Date and time that the Administrator last received a status update from the server. Like the Current Status, this information cannot be modified. The frequency of broadcasts is the one entered in the Administrator Configuration document. |
| Log | Click the Log document icon (if available) to see the latest Log document for this server. |
| Server Name | Name of the LEI server given at install time. |
| Poll Interval | The interval at which the LEI server polls the Administrator database to see if there are any Activities that it needs to execute. |
| Maximum Number of Activities | The greatest number of concurrent Activities that the server will run. Once the server is running the maximum number of Activities, it postpones additional Activities until existing Activities are concluded. Zero stands for no maximum. |

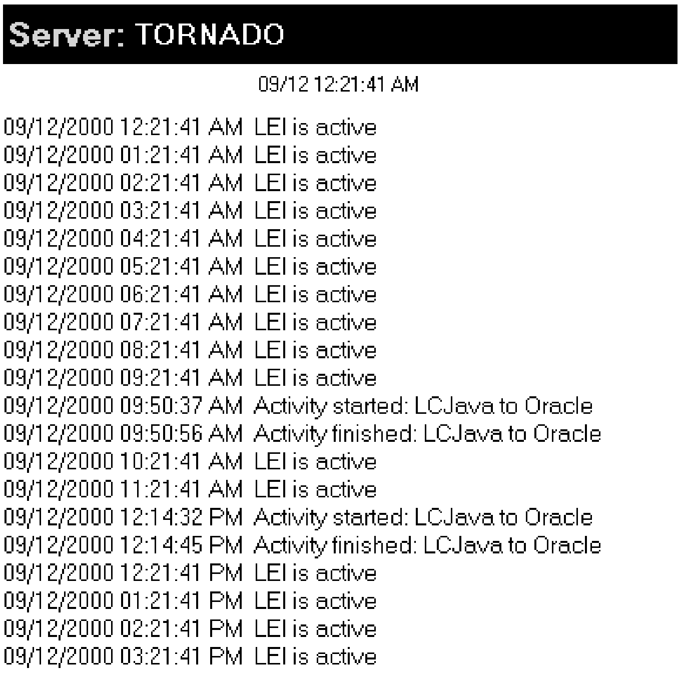
| | |
|---|---|
| Maximum Duration of Activities | The longest duration, in minutes, that the server will run any single Activity. The server automatically closes any Activity that exceeds this duration. Zero stands for no timeout. |
| Maximum Consecutive Failures | The greatest number of consecutive failures that a single Activity can have before the server no longer schedules it for execution. Zero stands for no disabling. |
| Activity Shutdown Request Timeout | The amount of time in seconds that an Activity has to respond to a Close command before its terminated. Zero stands for no timeout. |
| International Text Translation | This field can have one of three settings: Disabled, LMBCS Only, or Enabled.<br><br>Disabled – causes the LEI to assume all data is in compatible character sets and transfer the data without translation.<br><br>Enabled – causes the LEI server to automatically detect the database character set of the source and destination databases and perform translations of the data from source to target. Enabled text data translation causes LEI to perform all necessary character set translation during the transfer of data.<br><br>LMBCS Only – causes LEI to assume that all non-Notes data is in compatible character sets and only transfers to or from Notes will require translation.<br><br>International character sets supported by the LEI server are listed in Appendix C, "Character Sets." Support for over 200 different character mappings is provided, including US English, Asian, Far East, Western and Eastern European languages, as well as multi-lingual character sets such as LMBCS (Lotus Multi-byte Character Set) and Unicode. Using the LEI text translation capabilities, for example, an organization with business units in France and China may generate information in French at the source data server, and have it received appropriately in Chinese format by the destination data server located in China.<br><br>For installations where the Operating System's character set should be used regardless of the database character set, use the LMBCS Only setting. This will force all non-LMBCS character sets to be considered to be in the platform character set. |
| Category | This is a text field that allows you to enter a Category name of your choice. This allows you to group Activities by category. If the category you specify doesn't exist, it will automatically be created. You can enter multiple categories per document. |
| Comments | This is a rich text field (RTF), so you can place anything here, including links to documents. |

## LEI Log

LEI maintains a log for all LEI Servers and Activities. The LEI log is useful for tracking performance and for troubleshooting.

To view your LEI Log, select Log from the Administrator Navigator, then double-click the specific entry you want to look at.

Shown below is an example of an LEI Server Log entry. A Server Log entry contains the name of the LEI server, the server start and end times and dates, and all Activities run on the server. Start and finish times and dates are shown for each Activity.



```
Server: TORNADO
                    09/12 12:21:41 AM

09/12/2000 12:21:41 AM  LEI is active
09/12/2000 01:21:41 AM  LEI is active
09/12/2000 02:21:41 AM  LEI is active
09/12/2000 03:21:41 AM  LEI is active
09/12/2000 04:21:41 AM  LEI is active
09/12/2000 05:21:41 AM  LEI is active
09/12/2000 06:21:41 AM  LEI is active
09/12/2000 07:21:41 AM  LEI is active
09/12/2000 08:21:41 AM  LEI is active
09/12/2000 09:21:41 AM  LEI is active
09/12/2000 09:50:37 AM  Activity started: LCJava to Oracle
09/12/2000 09:50:56 AM  Activity finished: LCJava to Oracle
09/12/2000 10:21:41 AM  LEI is active
09/12/2000 11:21:41 AM  LEI is active
09/12/2000 12:14:32 PM  Activity started: LCJava to Oracle
09/12/2000 12:14:45 PM  Activity finished: LCJava to Oracle
09/12/2000 12:21:41 PM  LEI is active
09/12/2000 01:21:41 PM  LEI is active
09/12/2000 02:21:41 PM  LEI is active
09/12/2000 03:21:41 PM  LEI is active
```

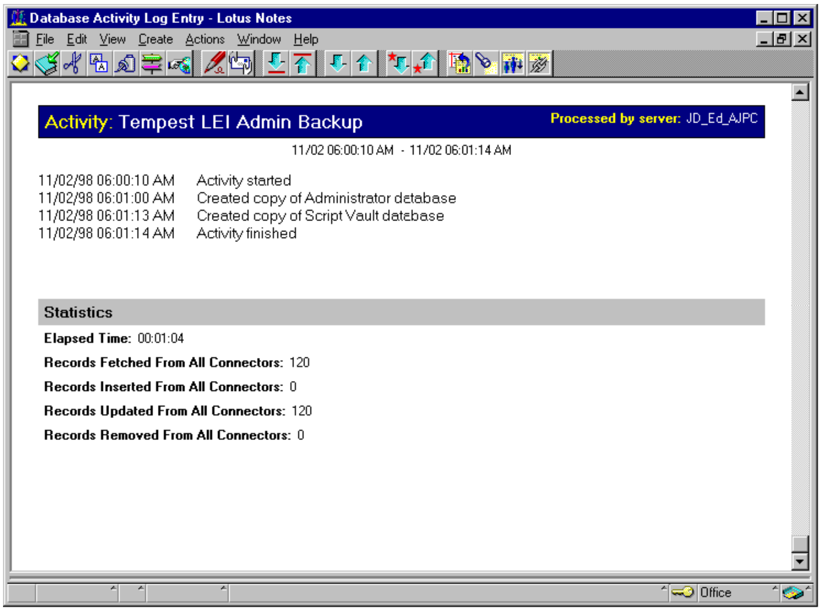
### Server Entry Information

The following information appears in the Server Log document:

| | |
|---|---|
| Server | Name of the server. |
| (Date/time interval) | The date and time that the server was started and terminated, marking the duration of the server's activity during this log entry. The date/time format depends on the current Notes client operating system. |
| Individual line items | History of processing undertaken by the server, including specific activities with their start and stop times, server status, and any error messages that may have occurred. |

## Activity Logs Entries

Shown below is an example of a LEI Activity Log entry. Activity Log entries provide information about LEI Activities that enable you to track system performance and results. The information in an Activity entry include the name of the Activity, the server that processed it, the start and finish date and time, and a summary of statistics.


Database Activity Log Entry - Lotus Notes
File Edit View Create Actions Window Help
Office

Activity: Tempest LEI Admin Backup

Processed by server: JD_Ed_AJPC

11/02 06:00:10 AM - 11/02 06:01:14 AM

```
11/02/98 06:00:10 AM    Activity started
11/02/98 06:01:00 AM    Created copy of Administrator database
11/02/98 06:01:13 AM    Created copy of Script Vault database
11/02/98 06:01:14 AM    Activity finished
```

Statistics

**Elapsed Time:** 00:01:04

**Records Fetched From All Connectors:** 120

**Records Inserted From All Connectors:** 0

**Records Updated From All Connectors:** 120

**Records Removed From All Connectors:** 0

### Activity Entry Information

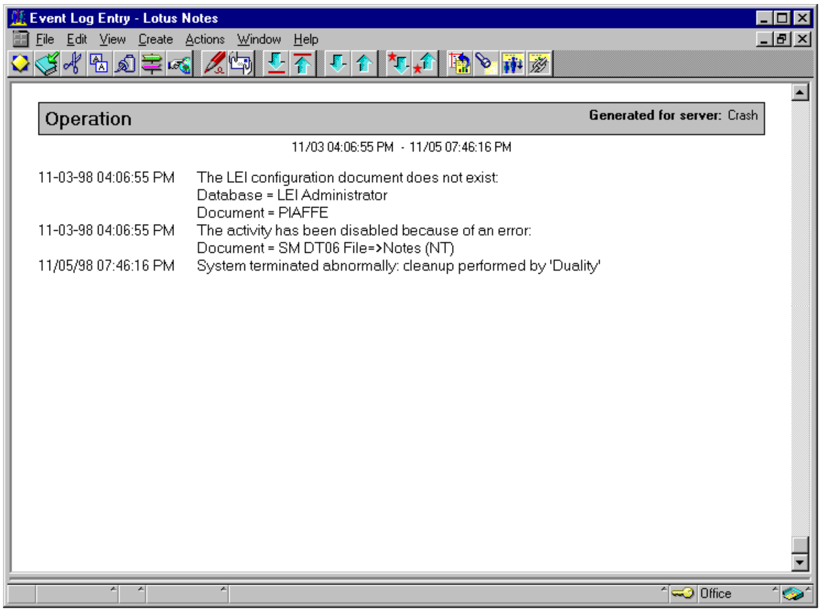
The following information appears in the Activity Log document:

| | |
|---|---|
| Activity | Name of the Activity. |
| Processed by Server | Name of the LEI that ran the Activity. |
| (Date/time interval) | The Activity's start and end date and time. The date/time format depends on the current Notes client operating system. |
| Individual line items | History of the Activity's processing, including event and error messages. |
| Elapsed Time | Duration of the Activity in the format hh:mm:ss. |
| Records Fetched | Number of records that were fetched from the external system. |
| Records Inserted | Number of records that were inserted into the external system. |
| Records Updated | Number of records that were updated into the external system. |
| Records Removed | Number of records that were deleted from the external system. |

## Operation Logs

The Operation Log holds any LEI system reports. LEI will only create one of these reports when an error has been detected within the LEI Administrator/Control Store. As an example, if the scheduling information for an Activity were corrupted, then an Operation Log would be created to point out the Activity and the affected data.

An example of an Operation Log is shown below.


Event Log Entry - Lotus Notes
File Edit View Create Actions Window Help
Office

Operation Generated for server: Crash

11/03 04:06:55 PM - 11/05 07:46:16 PM

```
11-03-98 04:06:55 PM    The LEI configuration document does not exist:
                        Database = LEI Administrator
                        Document = PIAFFE
11-03-98 04:06:55 PM    The activity has been disabled because of an error:
                        Document = SM DT06 File=>Notes (NT)
11/05/98 07:46:16 PM    System terminated abnormally: cleanup performed by 'Duality'
```

### Operation Log Information

The following information appears in the Operation Log document:

| | |
|---|---|
| Generated for Server | Name of the server for which the log was generated. |
| (Date/time interval) | The date and time that the events logged took place. |
| (Individual line items) | Operational problem. |

## LEI Script Vault

An optional database, the Script Vault database (leivlt.nsf) is supplied with LEI. It is empty upon installation. It can be used to store LotusScript Agents to be used in LEI. It contains an action called “Catalog All Agents” available at the View level which inventories the Agents and creates a “tracking” document for each. This document contains information about each Agent, such as name and creator, and allows the user to add additional comments. Each time the “Catalog All Agents” button is pressed, the agent tracking document is updated, new agents are cataloged and tracking documents that no longer have associated agents are deleted. It is suggested that this database be included in regular backup procedures.

## LEI Documentation Database

The *Lotus Enterprise Integrator User Guide* (leidoc.nsf) is provided as online help.

When you select Help in the LEI Administrator Navigator, the online version of the *Lotus Enterprise Integrator User Guide* is opened. The remainder of the LEI documentation set is supplied in .pdf or .nsf form on the LEI product CD-ROM, as well as in book form when you purchase LEI. See Chapter 1 for more information about the LEI documentation set and related product information.

# Chapter 5
# Introduction to Connectors

This chapter provides information about Connectors. Included is information about the different types of Connections, how to create them, and the applicable LEI Activities for each Connection.

## Connectors Overview

A Connection is a document defining how an LEI Server Connector will interact with a specific data source. The Connection provides specific access parameters, such as server names, user IDs and passwords. The “connected” databases are the databases that LEI can access with Connectors for executing Activities such as data transfers. Because the forms used with each Connection type (Lotus Notes, Oracle, DB2, etc.) are different, each Connection is explained separately.

Before creating Connections, you should verify that the LEI Server can communicate with your system databases by running the data source-specific tests described in the *LEI Domino Connectivity and Installation Guide* or in the specific manual for the Domino Connector that you intend to use.

After creating your Connections, you should test them using CONTEST or LCTEST, described later in this chapter.

## Connectors and Supported Data Sources

LEI provides the following standard Domino Connectors for the associated data sources:

- Notes
- DB2
- EDA/SQL
- File System
- ODBC
- Oracle 7
- Oracle 8
- OLE DB
- Sybase
- Text

Additional Domino Connectors are available and can be purchased separately. These include Domino Connectors for:

- JD Edwards OneWorld
- SAP R/3
- PeopleSoft
- Oracle Applications
- Open Components Extension for Lotus Domino (Lawson)

For more information about supported Connectors, see the Web site www.lotus.com/dominoei.

## Using Multiple Connections of the Same Type

You can use multiple Connections for the same types of databases and even for the same databases.

You can have multiple Connections to the same database by making copies of a Connection definition and then renaming the copies. You might do this is to access the same database but to set various permission levels for specific tables.

Any number of Activities can use the same Connection.

Some Connections are not supported by certain Activities. See the section entitled “Connections and Supported Activities” later in this chapter for more information.

### MetaConnections: Specialized Intermediate Connections

MetaConnections are specialized Connections that perform an intermediary role when used with an Activity. MetaConnections enable specific actions to be performed on data. Supported MetaConnections include the following:

- Collapse/Expand
- Connection Broker
- Metering
- Order
- Trace

### Processing Attachments Stored in RTF Fields

Notes is primarily a document management system, not a relational database management system (RDBMS). Because of this, Notes performs special handling for attachments stored in rich text fields. It stores the attachment data separately from the rich text field data. Currently you can only transfer attachments from Notes to Notes; rich text fields with attachments do not store correctly in the target RDBMS such as DB2 and Oracle.

To work around this constraint, you can use the Extract File Attachments option in the Notes Connector to save attachments to files on disk. You can then use LotusScript to read the files, store their contents in LCFields, and save the LCFields to binary fields in a RDBMS. This level of scripting requires LotusScript expertise.

## Processing Empty and NULL Values

Empty data in a Notes Rich Text or Text field is not set to NULL when it transferred. It is transferred as " ". To set empty data to NULL requires that you write a LotusScript routine or a stored procedure.

NULL is not equivalent to an empty string or to an empty value such as zero. LEI regards a NULL as an undefined value, one that cannot be compared to other values. LEI Replication occurs whenever LEI performs a comparison between Database A and Database B and encounters naming mismatches in the comparison fields. For example, a replication will occur if LEI finds a NULL in one database matching an empty value in another. This can occur frequently with Date specifications - for example, if Database A specifies a date and Database B has no data specification. When this occurs, LEI uses SQL syntax to convert NULLS to values that are equivalent across databases.

## Supported Characteristics with List of Terms

The following terminology list defines the terms used in the "Supported Characteristics" section seen in each Connector chapter in this manual.

| *Term* | *Definition* |
|---|---|
| Writeback support | Indicates whether the Connection supports writeback result sets.<br><br>The term writeback is defined in RDBMS systems as an *updateable cursor*, which means that, using the result set, the most recently fetched record can be directly updated or deleted. A non-writeback update must first search the table using keys, however a writeback update can immediately locate the record to update. Writeback support saves time and is therefore more efficient. |
| Writeback index | The database index, if any, required to enable writeback support for a given metadata object. For example, some Connectors require that a unique index exist, while others do not. |
| Statement syntax | The statement type and format that this Connector accepts. |
| Condition syntax | The type and format of the condition statements that this Connector accepts. |

*continued*

| *Term* | *Definition* |
|---|---|
| Array transfer | The capability of this Connector to transfer many records at once. The benefit of transferring many records at once is that only one network transaction is needed, rather than many, for a given set of records – the data has the same number of bytes, but the transfer takes less time. Array transfers can be used for read or write operations, depending on the RDBMS and the Connector. |
| Actions supported | Actions that this Connector is capable of supporting. |
| Catalog types | Catalog types that this Connector supports for browsing-type operations. |
| Create types | Object types that this Connector can create in the external system. |
| Drop types | Object types that this Connector can drop from the external system. |

## Connection Documents

Connection documents enable you to define your system Connectors. They contain access information for specific databases and users. When you create an Activity, you make Connection choices from the existing set of defined Connections specified in the Connection document.

The information required in a Connection document may be different for different database products, so the Connection documents provide standard Connection Name fields as well as sections and fields specific to the database. Refer to the specific Connection chapters in this manual for information about the type of Connection you are creating. If you are using a premium Connector, refer to the documentation provided with the Connector.

You can edit Connections for a particular Activity using the Edit Connection button.

### Creating a New Connection Document

To create a new Connection document, do the following steps.

1. Choose the Create Connection icon in the LEI Navigator.
2. Select the database type option from the menu.
3. Enter the required information in the Connection Definition form.

### Viewing Connection Documents

Once you create a Connection document, it is visible in the Connection view.

Select the Connections view from the LEI Administrator Navigator, or from the menu, choose View – Connection – By Name; or choose View – Connection – By Type.

## Features Common to All Connection Documents

The following sections describe the parts that are found in every Connection document. For information about the unique parts of each Connection document, refer to the appropriate Connection chapter.

#### Pop-up Help

Dark blue text in the section headings in a Connection document indicates that pop-up help is available. To display pop-up help, place the cursor on the heading and press and hold the mouse button.

#### Connection Name

This field provides a name that identifies the Connection. Each Connection requires a unique name.

#### Password Encryption

This option enables the author of a Connection document to encrypt passwords using Notes encryption keys. It requires that all encryption keys used to encrypt Connections be added to the LEI server's Notes User ID. You can encrypt the password in either the Activity or the Connection. However, password encryption in LEI pertains specifically to the password as stored in the Notes database and as seen on the screen, not for communicating to the backend database. When data is sent to the backend database, encryption is removed.

Password encryption allows LEI users to encrypt password information using their own encryption keys. The LEI server is always able to read the passwords.

To enable encryption for a Connection document, click the Encryption button in the Connection and then select the encryption key to use.

**To Add Encryption Keys**
Use the Notes Actions – User ID – Encryption Keys from the Notes menu to create Notes ID encryption keys. See the Encryption section of the Notes online help for more information.

## Calling External Procedures

Procedures in external systems may be used in a number of forms-based Activities and from within LEI LSX scripts. Some Connectors provide procedure functionality as an alternate method of performing data selection or modification operations, while for others it is the only method of communicating with an external system. In all cases, there are various contexts and requirements regarding the behavior and handling of parameters for procedures.

Instructions for using procedures within a given Activity can be found in the documentation for that Activity. Any special considerations for calling procedures within a specific Connector can be found in the documentation for that Connector. In general, parameters will be provided to procedures by name if named parameters are supported, otherwise by position. Output from procedures will be expected as returned result sets when supported, otherwise output parameters may be used.

A procedure is generally used within three general contexts:

- **Perform a black box action**.

  In this case, the caller is not concerned with the action being performed – inputs may be provided, but no specific behavior or output is expected. This context is generally used within an LEI LSX script, as most forms-based LEI Activities expect specific behavior from a stored procedure. In this case, input parameters are provided as indicated in the script, and parameters of the corresponding name and data type will be supplied to the procedure. For Connectors which do not support named parameters, the parameter position becomes relevant.

- **Perform a backend modification**.

  In this case, the stored procedure is being used to alter data in the external database. This context is used in many situations; both from forms-based Activities and scripts. When used from a script, it is treated like the black box action described above. When used from forms-based Activities, the input parameters provided and the behavior expected is defined by the normal Activity behavior. For example, when using a stored procedure as the destination of a Direct Transfer Activity, the input parameters match the destination fields being stored, and the insertion of new data records is the expected behavior. The Activity in question will specify what parameters are provided for a given context. For example, a single RealTime Notes Activity can perform inserts, updates, and deletes through different stored procedures, and the parameters provided are specific to the action being performed.

- **Produce a result set**.

  In this case, a procedure is being used to return information from an external system, either to simulate a standard selection operation or as the only method of producing a result set, depending on the Connector. Input parameters may be provided, and a result set should be returned. The input parameters are supplied in the same manner as the other procedure contexts described above. The result set contents are generally determined by the procedure definition. In some systems, the fields being returned must be identified at the time of the procedure call by the caller. In this case, the output fields will be identified by the calling Activity, or manually within the script. An example of this behavior is Oracle procedures, which only return results in output parameters that must be identified at the time of the call.

## Testing Connectivity to External Data Sources: LCTEST

LCTEST is a program that tests system connectivity to external data sources.

### Supported Data Sources

LCTEST tests system connectivity to the following supported data sources:

- Notes
- DB2
- EDA/SQL
- ODBC
- Oracle 7
- Oracle 8
- OLE DB
- Sybase

## Requirements

Before running LCTEST, you must have the appropriate software installed on the Domino host for each data source you want to test.

## LCTEST Syntax

The syntax for using LCTEST is as follows:

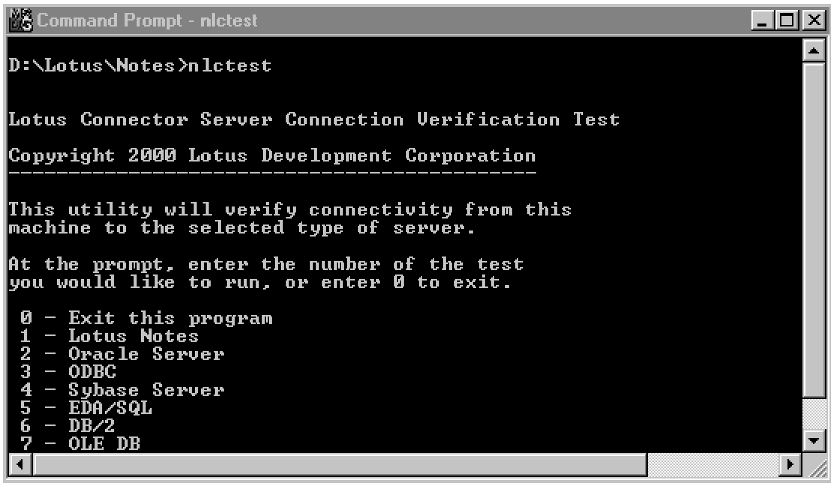
```
[n]lctest
```

## Running LCTEST

Follow the steps below to run LCTEST.

1. Locate the LCTEST program specific to your operating system platform in the Domino program directory. The LCTEST program has the following names for each of the associated operating system platforms:
   - All Windows (Win32) Platforms – nlctest.exe
   - All UNIX Platforms – lctest

   **Note** Because UNIX is case-sensitive, be sure to enter the LCTEST program name using lower case characters.
2. Double-click the program name to launch it, or type the program name at the system prompt. The LCTEST screen appears, as shown below.



```
D:\Lotus\Notes>nlctest


Lotus Connector Server Connection Verification Test

Copyright 2000 Lotus Development Corporation
---------------------------------------------

This utility will verify connectivity from this
machine to the selected type of server.

At the prompt, enter the number of the test
you would like to run, or enter 0 to exit.

 0 - Exit this program
 1 - Lotus Notes
 2 - Oracle Server
 3 - ODBC
 4 - Sybase Server
 5 - EDA/SQL
 6 - DB/2
 7 - OLE DB
```

3. Enter the number of the test you want to run and press Enter. Depending on the type of data source you are testing, you are prompted to enter additional information to log in to the specified data source. For more information, refer to the *LEI Domino Connectivity and Installation Guide.*

**Note** Testing an ODBC Connection with LCTEST can provide you with information regarding the minimum version and conformance level required of ODBC drivers so that they work with LEI. However, to obtain this information you must respond with Yes (Y) when LCTEST prompts you for a full report.

## Testing Connections: CONTEST

CONTEST is an additional testing program, similar in concept to the Connector-specific test program LCTEST. CONTEST must be run with a running LEI server. CONTEST attempts to connect using Connections defined in the currently running LEI server's Administrator database. CONTEST tests the ability to make a Connection through the information found in the Connection document.

### CONTEST Syntax

The syntax for using CONTEST is as follows:

```
[n]contest [Options] <Connector1> <Connector2>...<Connectorn>
```

**[Options]** – denotes the optional parameter: -p. This parameter displays the Connector properties.

**<...>** – denotes actual Connection names (use quotation marks if spaces exist in the Connection names).

**Note** Typing [n]contest with no input parameters results in Help information being displayed.

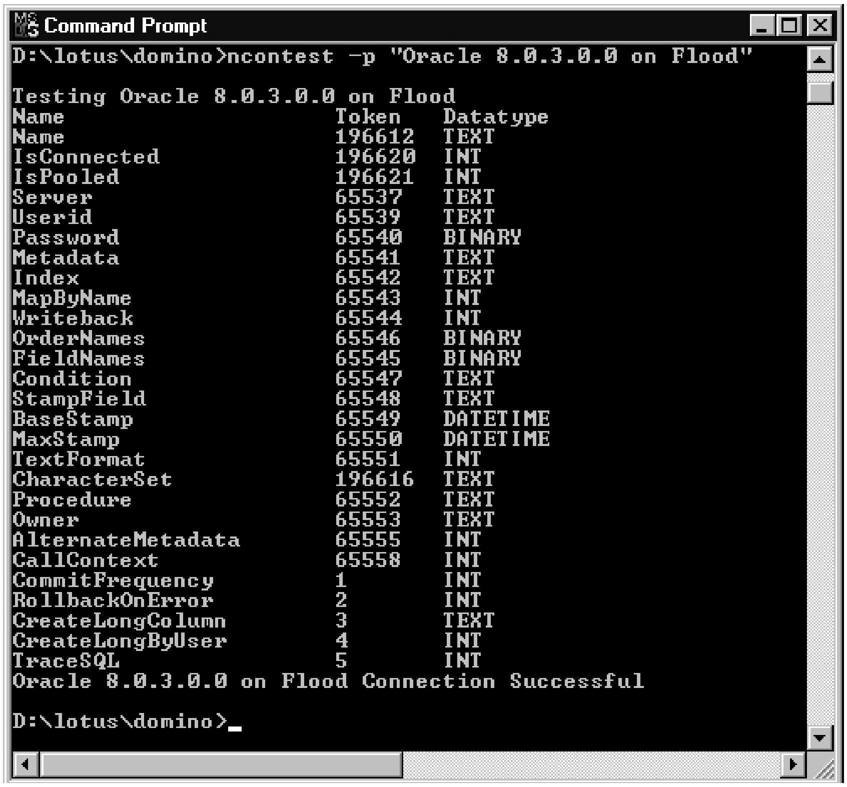
### Running CONTEST

Follow the steps below to run CONTEST.

1. Locate the CONTEST program specific to your operating system in the Domino program directory. The CONTEST program has the following names for each of the associated operating system platforms:
   - All Windows (Win32) Platforms – ncontest.exe
   - All UNIX Platforms – contest

   **Note** Because UNIX is case-sensitive, be sure to enter the CONTEST program name using lower case characters when working on a UNIX platform.
2. Type the program name at the system prompt. CONTEST runs in a command window.

**Note** Information about the Connection displays when you use the optional parameter –p. Also, when you enter [n]contest without the -p parameter, only information about the successfulness of making the Connection displays.



```
D:\lotus\domino>ncontest -p "Oracle 8.0.3.0.0 on Flood"

Testing Oracle 8.0.3.0.0 on Flood
Name                        Token    Datatype
Name                        196612   TEXT
IsConnected                 196620   INT
IsPooled                    196621   INT
Server                      65537    TEXT
Userid                      65539    TEXT
Password                    65540    BINARY
Metadata                    65541    TEXT
Index                       65542    TEXT
MapByName                   65543    INT
Writeback                   65544    INT
OrderNames                  65546    BINARY
FieldNames                  65545    BINARY
Condition                   65547    TEXT
StampField                  65548    TEXT
BaseStamp                   65549    DATETIME
MaxStamp                    65550    DATETIME
TextFormat                  65551    INT
CharacterSet                196616   TEXT
Procedure                   65552    TEXT
Owner                       65553    TEXT
AlternateMetadata           65555    INT
CallContext                 65558    INT
CommitFrequency             1        INT
RollbackOnError             2        INT
CreateLongColumn            3        TEXT
CreateLongByUser            4        INT
TraceSQL                    5        INT
Oracle 8.0.3.0.0 on Flood Connection Successful

D:\lotus\domino>_
```

## Examples using CONTEST on an NT Platform

Example 1:

```
C:\LEI>contest "NotesDb on MyOracle"
```

Sample Output:

```
Testing NotesDb on MyOracle

dummy1 on MyOracle Connect Successful
```

Example 2:

```
C:\LEI>contest "Oracle on MyOracle " "Sybase 11 on
MySQLServer"
```

Sample Output:

```
Testing Oracle on MyOracle

Oracle on MyOracle Connect Successful

Testing Sybase 11 on MySQLServer

Sybase 11 on MySQLServer Connect Successful
```

## Connectors and Supported LEI Activities

The table below shows which Connectors work with each LEI Activity.

| *Connector* | *Archive* | *Command* | *Direct Transfer* | *Java* | *Polling* | *RealTime Notes* | *Replication* | *Scripted Activities* |
|---|---|---|---|---|---|---|---|---|
| **DB2** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **Oracle 7** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **Sybase** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **Notes** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **ODBC** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **Text** | Yes | Yes | Yes | Yes | No | No | No | Yes |
| **EDA (read + insert only)** | No | Yes | Yes | Yes | Yes | Yes | Yes (source only) | Yes |
| **File** | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes |
| **OLE DB** | Yes | Yes | Yes | No | No | Yes | Yes | Yes |
| **Oracle 8** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **No Connector** | No | Yes (for OS) | No | N/A | No | No | No | N/A |

**Note** The Command Activity is the only Activity that does not require a Connection. In such a case, the Command Activity would be executing an operating system command on the system on which the LEI Server is running.

## Note on Connector Data Type Conversion Tables

This section describes how LEI handles data type conversions. It is provided as background information. This information may be particularly helpful if you are scripting with LEI LSX or the LC Java Classes.

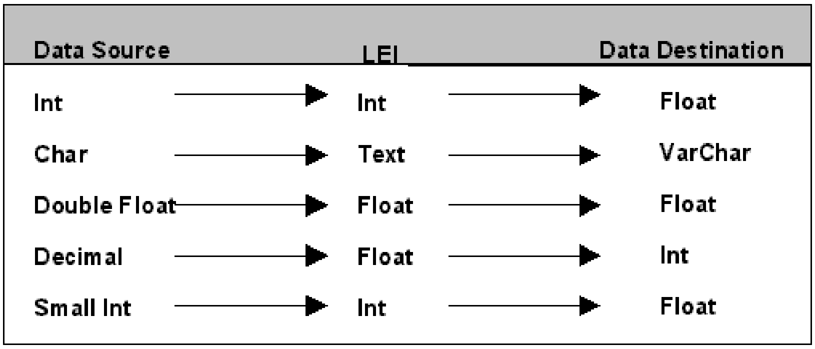
At the end of each Connector chapter is a section that includes tables that define data type conversions for the specific data source defined by the Connector. These tables arrange LEI transfers into four sections: Execute, Fetch, Insert/Update, and Create.

**Note** The Execute and Create tables are internal tables. The LEI data types described in this section are internal data types.

During execution of an Activity, data is transferred from its native data type into an LEI field object, which is a component of a Fieldlist object. A Fieldlist is a collection of named fields which is temporarily stored (in memory only; not on disk) on the LEI server. The data is then transferred, replicated, copied, etc., to the destination in the most compatible data type that LEI can determine. So data can undergo two changes: first into an LEI field, and then into the compatible target format.

Conversions such as character set are done at the last minute, and directly from source to target to avoid extra or unnecessary conversions.

The figure below illustrates the conversion process of data types. Note that this is meant to show how data types might be converted during an LEI Activity. For information about the specific conversions of data types for specific data sources, refer to the data type conversion tables in the Connector chapters of this manual.



The information below provides details on the data type conversions.

- **Execute** operations against a LEI Connection produce a result set from the data source and a corresponding metadata structure called a Fieldlist in LEI. The Execute table defines what type of LEI field will be added to a Fieldlist based on the type of the field in the result set. In general, each data source type maps to a single LEI type. Any multiple choices indicate that certain attributes of the source data type can alter the LEI type used.
- **Fetch** operations define what data transformations are valid in fetches from a data source result set into an LEI Fieldlist. In a Direct Transfer and most forms-based Activities, the Fieldlist produced by a Connector in Fetch is sent back to that Connector for Fetch, but from an LEI script there is no such restriction. For each LEI data type, the table indicates which data source columns may be fetched into that field. Multiple choices indicate that various data source types will be converted to the indicated LEI type. For an LEI binary field, the different formats allow for different type checking restrictions (for example, an LEI number list allows numbers to be fetched from Notes, but an LEI datetime list does not).
- **Insert/Update** operations are the opposite of a Fetch operation, and for each LEI data type the table defines the valid data target types which can be stored. Multiple values indicate that a single LEI type can map to multiple target types. Additional formatting information in a middle column has the same usage as for the Fetch operation above.
- **Create** operations create metadata in the data target from an LEI Fieldlist. The table defines which data target type is created based on the LEI data type. Multiple values indicate that formatting information from the LEI field affects the choice. This is especially relevant for Create operations since LEI fields can hold additional formatting information specifically to refine creations. This is most apparent during an operation like Direct Transfer, but much of this is invisible to the LEI user except when using advanced scripts. Execute operations store additional formatting information in the LEI Fieldlist metadata, which is then used during creation to exactly reproduce (as best supported by the data target) the source. For example, a 2-byte unsigned integer in a data source will be represented by the LEI generic INT type. But the original information is still stored in LEI, and for a data target which supports such a type, this information will be used during Create operations.

To summarize, the Execute and Create tables define the default mapping performed by LEI when creating LEI metadata from a data source, or data target metadata from an LEI Fieldlist. The Fetch and Insert/Update tables define valid data type mapping between LEI and the data source/target.

# Chapter 6
# Lotus Notes Connector

This chapter provides information about the Notes Connector, including descriptions of the fields in the Connection document.

## Introduction to the Notes Connector

The Notes Connector allows you to define a Connection to a Lotus Notes database.

## Terminology

The table below lists LEI terms and the corresponding terms in Notes.

| *LEI* | *Notes* |
|---|---|
| Server | Server |
| Database | Database |
| Metadata | Form |
| Index | View |
| Field | Field/Item |
| Record | Document |

## Supported Characteristics

The following list captures supported characteristics of the Notes Connector.

For a definition of the terms used in this section, see “Supported Connector Characteristics – List of Terms” in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Domino Connector LotusScript Extensions Guide* for more information.

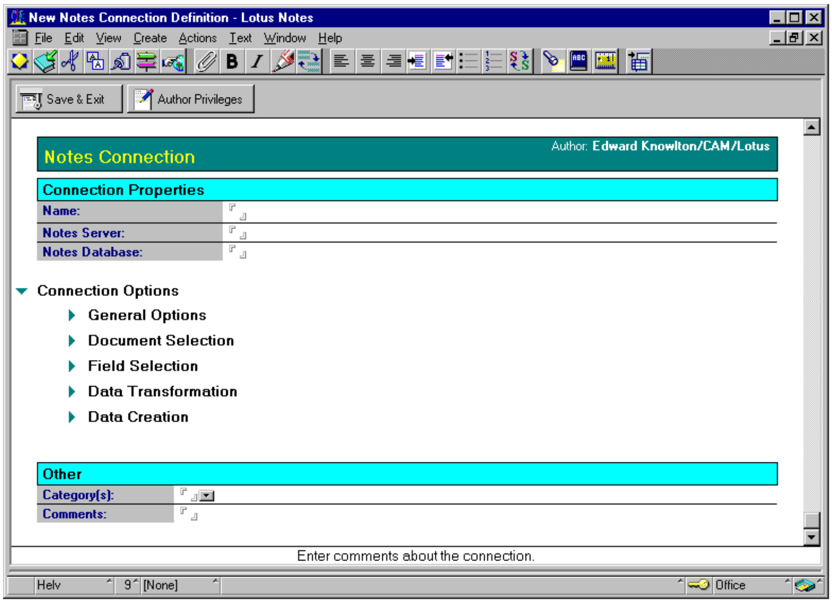
- Writeback support – Full
- Writeback index – When an OrderList is provided, it uses an exiting view, creates a temporary or permanent view LEIIndexViewXXX, where XXX is the first available number. Notes Connector properties are used to control view creation.
- Statement syntax – Notes formula language. Excludes data modification statements but includes “execute <agent name>” where <agent name> is a Notes agent that is not run on a view.
- Condition syntax – Valid when placed in the following Notes formula: SELECT <condition> …
- Array transfer – None (Bulk Insert)
- Actions supported – Clear, Reset, Truncate
- Catalog types – Server, Database (no connection required), Metadata, Index, Field
- Create types – Database (cannot be connected), Metadata, Index
- Drop types – Database (cannot be connected), Metadata, Index

**Note** All connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections “Common Connector Properties” and “LCConnection Properties” in the LCConnection Class chapter of that same manual.

## Notes Connection Document

The Notes Connection Document, shown below, defines a Connection to a Notes database and the Notes server on which it can be found.

Click the Create Connection icon in the Navigator or choose Create – Connection – Notes in the LEI Administrator to access the Notes Connection Document. The fields in the Notes Connection Document are described in the table below.



## Connection Properties

| *Field* | *Description* |
|---|---|
| Name | Enter a unique name for this Connection. |
| Notes Server | Enter the name of the server where the Notes database identified by this Connection is located. Leave this field blank if the database is to accessed locally. |
| Notes Database | Enter the file path and filename (*.nsf) of the Notes database identified by this Connection, relative to the Notes Data directory. For example, to access c:\notes\data\orders.nsf, enter orders.nsf in this field. |

## Connection Options

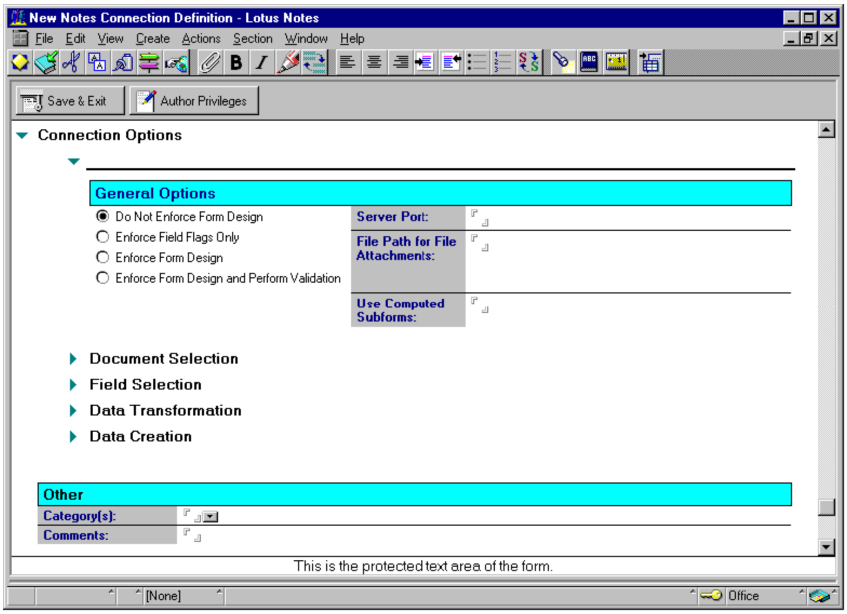
These advanced options extend the capabilities of LEI to take advantage of specific Notes data structures and design elements. The options let you specify, for example, view-based selection criteria that manage the hierarchical structure of Notes. Connection Options also permit you to define data translation and manipulation. The options are divided into the following groups:

- General Options (source and destination)
- Document Selection
- Field Selection
- Destination Column Data Types
- Data Transformation
- Data Creation

The fields in each of these sections are described in the corresponding sections below.

### General Options

These options can apply to the source or destination database, depending on whether the Connection is used by the Activity as the source or target Connection.


New Notes Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

Connection Options

General Options

[✓ Do Not Enforce Form Design] Enforce Field Flags Only / Enforce Form Design / Enforce Form Design and Perform Validation

| | |
|---|---|
| Server Port: | |
| File Path for File Attachments: | |
| Use Computed Subforms: | |

Document Selection
Field Selection
Data Transformation
Data Creation

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

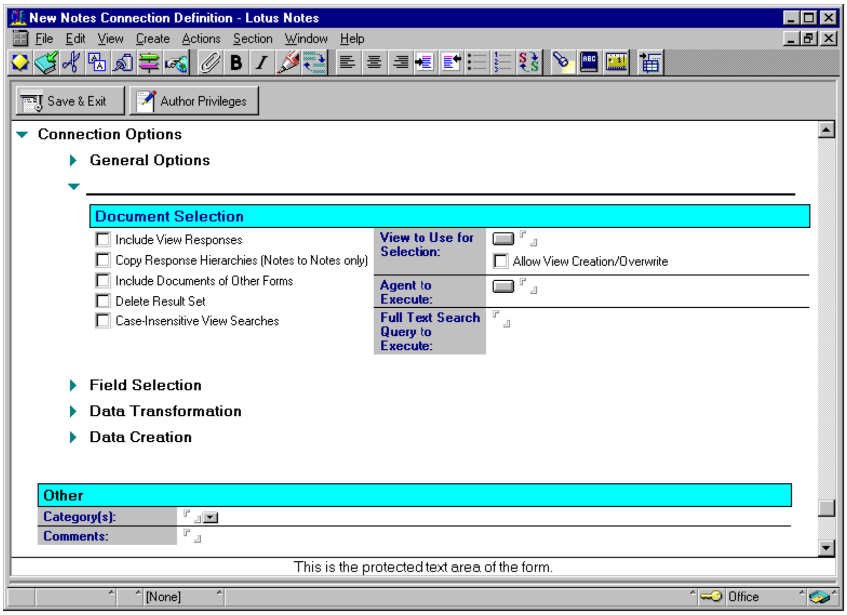
This is the protected text area of the form.

| *Field* | *Description* |
|---|---|
| Do Not Enforce Form Design – Default Option | If this option is selected, the Activity does not execute any Notes form- and field-related formulas. Computed fields (such as the current date) are not computed. |
| Enforce Field Flags Only | If this option is selected, the Activity sets all field flags as defined in the form but does not execute any formulas. As a result, special types, such as reader and author names, are assigned in new documents as indicated in the form. This option avoids the overhead of computing field formulas. |
| Enforce Form Design | This option calculates all field-related formulas (default value, input translation, and computed field) as part of the Activity. This option may cause slower data transfer as a result of formula calculations. This option does not execute @ function names that begin with Db. It does enable field flags. |
| Enforce Form Design and Perform Validation | This option calculates all Notes field-related formulas (default value, input translation, and computed field), and invokes all Notes field validation formulas which ensure that the data is within the parameters for the field in the Notes form. Any field validation errors are recorded in the Activity log. |
| Server Port | Name of the port to use when dialing a server remotely. |
| File Path for File Attachments | Specifies the directory where file attachments are stored if the option Extract File Attachments is selected (see below). |
| Use Computed Subforms | To support computed subforms, list all of the subforms to include here. All fields in these subforms will be considered to be part of the form. |

**Note** No LotusScript is executed by any of the above options. Only language formulas are executed.

## Document Selection

The Document Selection section is used to configure the document level options used by the Activity. These are used to limit the documents that the Activity selects from the source. Several options can be combined, but the filtering is cumulative. A document must satisfy all options to be selected. These options can apply to the source or destination database, depending on whether the Connection is used by the Activity as the source or target Connection, and apply to the creation of a result set. The fields in this section are described in the table below the following graphic.


New Notes Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

Connection Options

General Options

Document Selection

☐ Include View Responses
☐ Copy Response Hierarchies (Notes to Notes only)
☐ Include Documents of Other Forms
☐ Delete Result Set
☐ Case-Insensitive View Searches

View to Use for Selection:
☐ Allow View Creation/Overwrite

Agent to Execute:

Full Text Search Query to Execute:

Field Selection

Data Transformation

Data Creation

Other

Category(s):

Comments:

This is the protected text area of the form.

[None]

Office

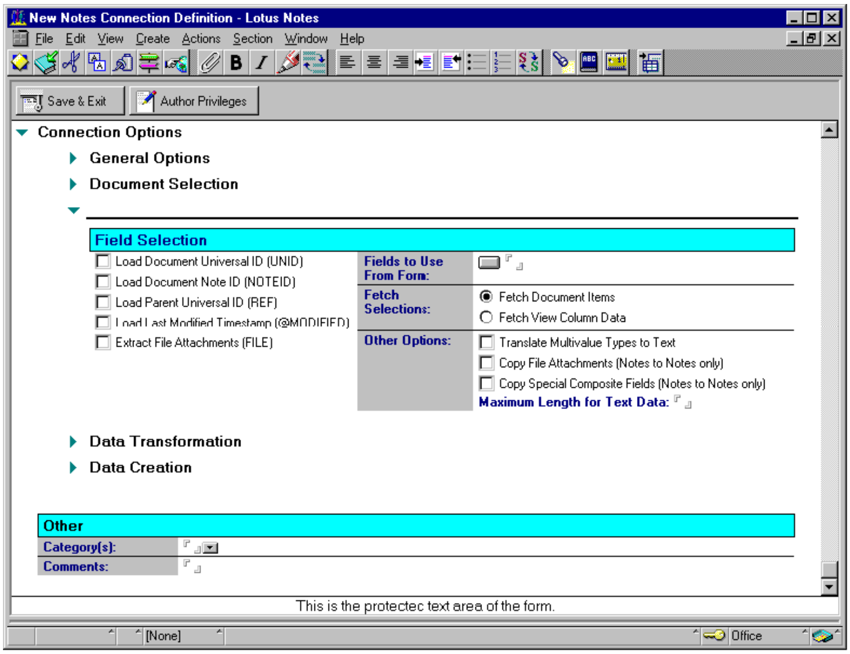
| *Field* | *Description* |
|---|---|
| Include View Responses | Includes response documents of any parent document that is used in the view. This option is relevant only for view-based result sets and cannot be used along with the option Copy Response Hierarchies.<br>When used in a Notes to Notes transfer, the result in the destination database is not hierarchical and all documents are copied at the same level. |
| Copy Response Hierarchies (Notes to Notes only) | This option ensures that response documents retain their hierarchical relations to the parent document. This option is relevant only for view-based result sets and cannot be used with the option Include View Responses.<br>To use this option from an Activity with manual field mapping, add a field LCXFILE to both field lists (this is not necessary for automatic field mapping by name or position). |
| Include Documents of Other Forms | By default, the documents in a result set are restricted to those associated with the source form identified as the metadata in the Activity document. Use this option to include documents of all forms in the result set, filtering them through the source form.<br>Note that if the option Fetch View Column Data is used for selection, this option is required and will be automatically selected. |
| Delete Result Set | Deletes documents from the source database before transfer. As a result, the documents don't show up on the destination database and are deleted from the source. To delete the source result set after a transfer, follow a regular transfer Activity with an Activity that deletes the result set. Identify the second Activity as a dependent Activity of the first.<br>This option only affects document selection through an explicit Notes formula, such as a Direct Transfer Activity or the LEI LSX Connection.Execute method. It does not affect generic selection such as done by Replication and RealTime Notes Activities.<br>This option applies only to the source database. It has no effect if applied to the destination. |
| Case-Insensitive View Searches | Makes view searches case-insensitive (they are case-sensitive by default). An example of where to use this is in timestamp replication with the case-insensitive string comparison option, where case-insensitive searches are needed in Notes. |

*continued*

| *Field* | *Description* |
|---|---|
| View to Use for Selection | If the name of a Notes view is entered, the documents appearing in that view will be selected by the Activity. This option replaces and ignores any command statement specified in the Activity. By default, only top-level documents are included, although other options allow the view contents to be altered. Documents not in the view are ignored.<br><br>When used with ordering or timestamp in Replication Activities, this field indicates the name of the view to use for these operations. When doing primary key replication, supplying a view name can improve performance by not requiring the database to be re-indexed with every Activity run. When no view is specified, a temporary view is still created and an event is logged suggesting use of a permanent view name.<br><br>**Note** When the view is the wrong format or doesn't exist, checking the Allow View Creation/Modification option (see below) allows LEI to create or overwrite that view in its own format. This option completely changes the view when modifying an existing view. |
| Allow View Creation/ Modification | Checking this option (when a view is specified) allows LEI to modify or create the view if it doesn't already exist or doesn't have the proper formatting. This option is required for timestamp replication when a view is specified. If the option "case-insensitive view searches" is selected, then the view columns will be set to sort case-insensitive.<br><br>When you use a permanent view to support keyed searches, such as those used by Replication Activities, and an existing view needs to be altered (which is always the case for timestamp Replication), it will be deleted and a new view will be created. |
| Agent to Execute | The agent specified will further refine the documents selected. The agent is executed on the result set produced by either the command statement in the Activity or the View to Use for Selection. |
| Full Text Search Query to Execute | The full text query you enter here is executed against the current result set and only the resulting documents (those that contain the searched text) are used by the Activity. The query is executed on the result set produced by either the command statement in the Activity or the View to Use for Selection, plus any defined Agent to Execute. The database must be indexed to support this option.<br><br>This option only affects document selection through an explicit Notes formula, such as a Direct Transfer Activity or the LEI LSX Connection.Execute method. It does not affect generic selection such as those done by Replication and RealTime Notes Activities. |

## Field Selection

The Field Selection section is used to limit the document fields (or view columns) used by the Activity to create the result set. The fields in this section are described in the table below the following graphic.



Field Selection

- [ ] Load Document Universal ID (UNID)
- [ ] Load Document Note ID (NOTEID)
- [ ] Load Parent Universal ID (REF)
- [ ] Load Last Modified Timestamp (@MODIFIED)
- [ ] Extract File Attachments (FILE)

Fields to Use From Form:

Fetch Selections: [● Fetch Document Items] ○ Fetch View Column Data

Other Options:
- [ ] Translate Multivalue Types to Text
- [ ] Copy File Attachments (Notes to Notes only)
- [ ] Copy Special Composite Fields (Notes to Notes only)

Maximum Length for Text Data:

Other

Category(s):

Comments:

This is the protectec text area of the form.

[None] Office

| *Field* | *Description* |
|---|---|
| Load Document Universal ID (UNID) | Appends a virtual field named UNID to the result set. This field will fetch the Universal Note ID for each document. See the section “Destination Column Data Types” below for more details. |
| Load Document Notes ID (NOTEID) | Appends a virtual field named NOTEID to the result set. This field will fetch the Note ID for each document. See the section “Destination Column Data Types” below for more details. |
| Load Parent Universal ID (REF) | Appends a virtual field named REF to the result set. This field will fetch each document’s parent’s Universal Note ID, which is used by Notes to maintain response hierarchies. See the section “Destination Column Data Types” below for more details. |

*continued*

| *Field* | *Description* |
|---|---|
| Load Last Modified Timestamp (@Modified) | Adds a field "@Modified" to the result set, which contains the Notes implicit document timestamp. Select this option on the relevant Notes Connection in conjunction with the Static Timestamp option in the Replication Activity to use Notes implicit timestamps in replication (no special timestamp field needs to be used). See the section "Destination Column Data Types" below for more details. |
| Extract File Attachments (FILE) | Appends a virtual field named FILE to the result set. FILE contains file names of all attachments in a document and extracts files to disk. The directory for file extraction is the current directory or the one specified in the File Path for File Attachments field in the General Options section of the Notes Connection document. While this option copies file attachments on a Notes to Notes transfer, the Copy File Attachments option is more efficient for this purpose because it does not write to files. See the section "Destination Column Data Types" below for more details. |
| Fields to use from Form | Enter the field names of the fields you want to use from the form. To select all fields, leave this field blank. |
| Fetch Document Items – Default Option | This option gets the field names from the form designated as source metadata. |
| Fetch View Column Data | This option gets data from the view designated at View to Use for Selection. The fields are automatically named column1, column2, and so on. The destination metadata must contain fields of this name, or use map by position or user defined mapping to ensure there are no field mapping failures.<br><br>Checking this option automatically checks the option Include Documents of Other Forms. Note that this option retrieves text data only. |
| Translate Multivalue Types to Text | Transfers the contents of multivalue types Text List, Number List, and Datetime List as Text. Useful when the destination database cannot accept the multivalue types. |
| Copy File Attachments (Notes to Notes only) | Copies any file attachments associated with transferred documents. If this option is not selected, the attachment icons will still appear in the transferred documents but with no attachments.<br><br>To use this option from an Activity with manual field mapping, add a field LCXFILE to both field lists (this is not necessary for automatic field mapping by name or position). |

*continued*

| *Field* | *Description* |
|---|---|
| Copy Special Composite Fields (Notes to Notes only) | Copies any composite support fields associated with transferred documents. This includes the fields, links, and fonts to return full fidelity of composed data.<br><br>To use this option from an Activity with manual field mapping, add a field LCXFILE to both field lists (this is not necessary for automatic field mapping by name or position). |
| Maximum Length for Text Data | Signals LEI to use this length as the maximum for Text type fields. This option is often used when a metadata is being created in a relational database and text lengths and types are important. The default text length is slightly under 64Kb.<br><br>If the Truncate Data When Necessary option is set in the Activity document, LEI truncates the text field to the maximum length indicated here before transfer. If the Truncate Data option is not checked, any record with one or more fields greater than this limit is not transferred, and LEI logs the error "Data overflow." |

## Destination Column Data Types

To use any of these options with transfers to a database other than Notes, you must already have columns available in the destination metadata. The columns are added as the last columns in the metadata (one for each of the fields in the order listed above).

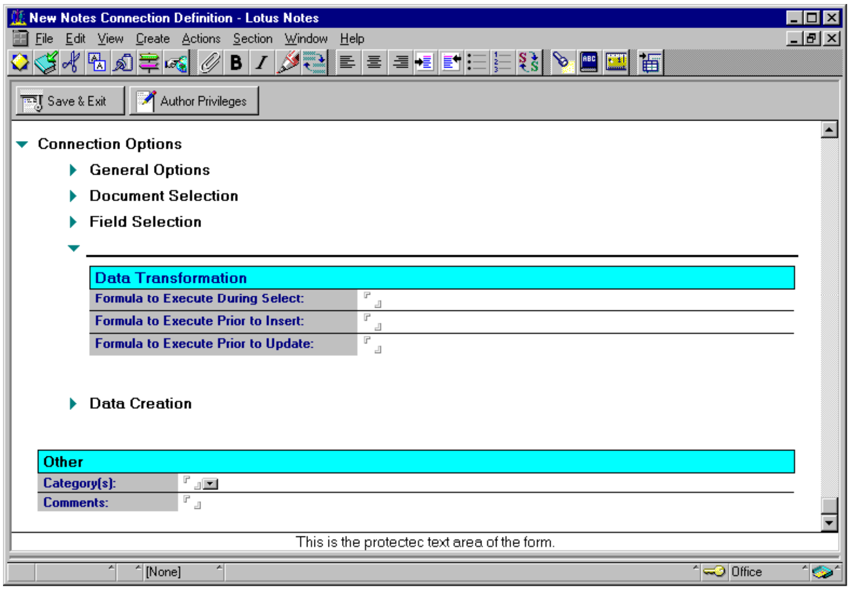
Some Notes Connection options add a special field to the result set. This field is automatically handled by a Notes target, but must be explicitly defined in a non-Notes target. When using LEI Replication between two Notes databases, these options must be selected in parallel, for both the source and target connections, to properly map within the Replication Activity.

The destination database column types must reflect the following Notes data types:

| | |
|---|---|
| UC | Binary BLOB (16) |
| NOTEID | Int |
| REF | Binary BLOB (16) |
| FILE | Binary Text List |
| @Modified | Datetime |

## Data Transformation

The Data Transformation section provides options for executing formulas to transform data.


New Notes Connection Definition - Lotus Notes

File Edit View Create Actions Section Window Help

Save & Exit | Author Privileges

Connection Options
General Options
Document Selection
Field Selection

Data Transformation

| | |
|---|---|
| Formula to Execute During Select: | |
| Formula to Execute Prior to Insert: | |
| Formula to Execute Prior to Update: | |

Data Creation

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

This is the protectec text area of the form.

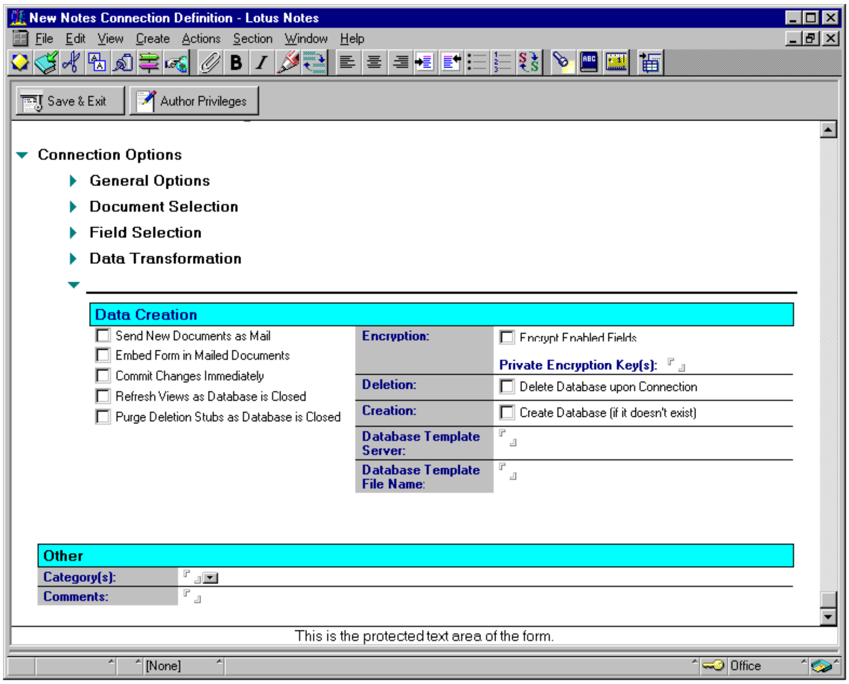
[None] Office

| *Field* | *Description* |
|---|---|
| Formula to Execute During Select | Enter the Notes formula that will execute just after the data is fetched. The data is altered as part of the fetch/select operation. |
| Formula to Execute Prior to Insert | Enter the Notes formula that will execute just before the selected data is inserted into the target. The data is changed as part of the insert operation. |
| Formula to Execute Prior to Update | Enter the Notes formula that will execute before the data in the designated target is updated. The transformation will take place as part of the update operation. |

**Note** Transformation formulas should be prefixed with “select @all;”. For example, to assign a field value, enter the formula <select @all; FIELD NewField := “New Value”;>.

## Data Creation

The Data Creation section provides options for managing data creation with the Notes database during processing of any Activities associated with the Connection.


New Notes Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
[None]
Office

Connection Options

- General Options
- Document Selection
- Field Selection
- Data Transformation

**Data Creation**

- ☐ Send New Documents as Mail
- ☐ Embed Form in Mailed Documents
- ☐ Commit Changes Immediately
- ☐ Refresh Views as Database is Closed
- ☐ Purge Deletion Stubs as Database is Closed

| | |
|---|---|
| Encryption: | ☐ Encrypt Enabled Fields<br>Private Encryption Key(s): |
| Deletion: | ☐ Delete Database upon Connection |
| Creation: | ☐ Create Database (if it doesn't exist) |
| Database Template Server: | |
| Database Template File Name: | |

**Other**

| | |
|---|---|
| Category(s): | |
| Comments: | |

This is the protected text area of the form.

| *Field* | *Description* |
|---|---|
| Send New Documents as Mail | Sends inserted documents as mail rather than saving to a Notes database.<br><br>For this option to work, the transferred records must contain a field named SendTo containing the valid name of the addressees. For sources that support SQL, this field can be added as part of the SQL SELECT clause in the statement of the Activity document. For example: “SELECT column1, column2, SendTo=“Phineas Seattle/Chief” FROM table.” |
| Embed Form in Mailed Documents | Embeds the form in documents sent as a result of using the Send New Documents as Mail option. Use this option when the recipients of the mail will not have the relevant form in their mail database. |
| Commit Changes Immediately | This option commits each change to the Notes database immediately. The default behavior is to commit all changes at database close. This option can cause slow performance. |

*continued*

| *Field* | *Description* |
|---|---|
| Refresh Views as Database is Closed | Reflects the database changes in the database's views at the end of the transfer or replication process when the destination database is being closed. Since views can be slow to recompute following many changes, this option causes the refresh to occur as part of the transfer rather than when the user next opens the view. |
| Purge Deletion Stubs as Database is Closed | Use this option to clear all deletion stubs from the Notes database when disconnecting. It is strongly recommended that this only be used with the option to "Refresh Views as Database is Closed", since once the stubs are deleted, they will not be properly removed from views. But selecting the Refresh Views option will remove them before purging. This option cannot be used on a Notes database with the Notes replication setting to "Delete documents not modified in the last N days." Doing so will skip the purging and generate an error indicating the conflict.<br><br>To use this option successfully, the LEI Notes ID must have sufficient access to the database to enable modification of Notes Replication settings. |
| Encrypt Enabled Fields / Private Encryption Keys | The option Encrypt Enabled Fields encrypts all encryption-enabled fields in the form of the target document. To use private encryption keys (the default is to use the public key), provide one or more private encryption key names, separated by commas or semicolons, in the Private Encryption Keys field that appears when you enable this option. These keys must be available on the LEI Server. This second option is valid only if the first is True. |
| Delete Database upon Connection | Deletes the database specified in the Activity upon connection. This is used together with Creation (below) to delete and then recreate the database prior to population. |
| Create Database (if it doesn't exist) | Creates a destination Notes database to receive the data if a database does not exist. For database creation, you can optionally enter a template server and template file path.<br><br>**Note** This option should not be selected when the target Domino server is part of a Domino cluster because the database may be created in an inconsistent state. When the target is a Domino cluster, create the database using a Notes client prior to running the LEI Activity. |
| Database Template Server | The Domino Server containing the template file to be used to create the destination database. |
| Database Template File Name | The file path of the Notes .ntf template file to be used to create the destination database. |

## Considerations when Working with Notes

This section addresses important issues in using a Notes Connection.

### Activity Command Statements

Aside from standard selection formulas using the Notes formula language, statements executed against Notes may also be of the following format:

```
"execute agent name"
```

where “agent name” is the name of a Notes agent.

This functionality can be used to modify data through a post-statement Polling Activity, where normally an update statement would be executed. By using this syntax, an agent that updates documents may be run. This is an LEI-specific formula language enhancement to account for that fact that non-selection formulas cannot be used as LEI statements.

The “execute <agent>” statement syntax requires a sufficient access level to perform the desired actions and access the agent.

The agent cannot be view-based, that is, specified as “run on selected documents in view” or “run on all documents in view.”

The agent must satisfy the following requirements:

- Shared agent
- Run manually from actions menu or manually from agent list
- Run on all documents in database or run once
- Do not use simple actions

**Note** Although the Activity document appears to support it, a Notes to Notes RealTime Activity is not supported.

### LEI Replication of Notes

When you make modifications to a client for Timestamp Replication, there is no way to obtain the server current datetime. Because of this, time differences between the Notes client and server are relevant. If the client is later than the server, then some changes may not be picked up until the next replication. If the client is earlier than the server, then some changes may be missed. If synchronization of time is not possible between client and server, then making the Domino Server slightly earlier than the clients will prevent documents from being missed.

When specifying a view to use (View property of Notes Connection Options), that view must be of a particular LEI format, and the keys indicated in the view are irrelevant (they are taken from the Activity's view keys, regardless of the Activity). If the view is not of the correct format, then it will be modified by LEI, using the AlterView property of the Notes Connection Options. When a view is specified and the view is not of the correct format, then it will be recreated for the proper result set. For this reason, do not use a view which is used for other purposes because it will be overwritten. When no view is specified, a temporary view is created for the duration of that Activity and a message is written to the log recommending that a permanent view be used. When using a view name with Timestamp Replication, the same view name is used between replications, but the view is still altered each time since it is categorized based on the last replication Timestamp.

## Notes Form Creation

Notes form and field attributes, such as field formulas, help text, and formatting, will be set as defaults when LEI creates a new Notes form.

## Notes @ Functions

The following @ commands cannot be used in any Notes formula specified in an Activity. They are not supported except through the Notes interface:

- @DbLookup
- @DbColumn
- @Command
- @MailSend
- @Prompt
- @DDEInitiate
- @DDETerminate
- @DDEExecute
- @DDEPoke

## Depositor Level Access

When the LEI Server connects with Depositor level access, documents in the database are not visible. Any selection results in no documents. One effect is that overwriting or truncating data will not find any documents and therefore not remove them.

### Moving Text List Data over Different Platforms

Notes databases store data in canonical format (byte ordering), which is platform-independent. LEI automatically converts data to the host platform's byte ordering when read from Notes, and to canonical format when written to Notes. (Composite data is an exception: It is always left in canonical format.) As a result, all Notes-format data outside of Notes is considered to be in platform byte ordering native to the host format, allowing users to ignore the implications of canonical byte ordering.

When moving data between Domino servers and Notes and other databases, there is normally not a problem. In the case, however, where data is moved from one Domino server to another Domino server through another database, and using LEI server platforms with different host formats, list types (text list, number list, and datetime list) become corrupted. For example, you use a Windows NT LEI server to move a text list from Notes to DB2. If the data is moved from DB2 back to Notes through another NT server, there is no problem. But if the other LEI server is a UNIX server, then the data is corrupted.

## Notes Data Types

The following tables show the correspondences of Notes to LEI data types for various types of Activities. The following notes apply to the tables below:

- (p) – Indicates that if Allow Precision Loss is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not an error is generated.
- Text, Number, or Time indicates single-value unless otherwise specified.
- The Notes Connection Option "Maximum Length for Text Data" alters the default boundary of 64996 for LEI text.
- The Notes Connection Option "Translate Multivalue Types to Text" makes multi-value Text, Data, and Time look like single-value Text.
- LEI evaluates the absence of a field in a Notes database as a NULL.

For more information on the organization of these tables, see "Note on Connection Data Type Conversion Tables" in Chapter 5.

## Execute

| *Notes* | | *LEI* |
|---|---|---|
| Number | Single-value – Notes Number field contains a single value | Float |
| | Allow multi-values | Binary (number list format, variable length, bound = 64996) |
| Time | Single-value | Datetime |
| | Allow multi-values | Binary (datetime list format, variable length, bound = 64996) |
| Keywords | | Text (variable length, bound = 64996) |
| Names | | Text (variable length, bound = 64996) |
| Author Names | | Text (variable length, bound = 64996) |
| Reader Names | | Text (variable length, bound = 64996) |
| Text | single-value | Text (variable length, bound = 64996) |
| | allow multi-values | Binary (text list format, variable length, bound = 64996) |
| Rich Text | | Binary (composite format, variable length, unbounded) |

## Fetch

| *LEI* | | *Notes* |
|---|---|---|
| Int | | Number (p, o) |
| Float | | Number |
| Currency | | Number (p, o) |
| Numeric | | Number (p, o) |
| Datetime | | Time |
| Text (o) | | Text, Rich Text, Formula |
| Binary (o) | BLOB | Text, Number allow multi-values, Time allow multi-values, Rich Text, Formula |
| | composite | Text (single-value), Rich Text |
| | number list | Number, Text (single-value) |
| | datetime list | Time, Text (single-value) |
| | text list | Text |

## Insert/Update

| *LEI* | | *Notes* |
|---|---|---|
| Int | | Number |
| Float | | Number |
| Currency | | Number (p) |
| Numeric | | Number (p) |
| Datetime | | Time |
| Text | | Text, Rich Text, Formula |
| Binary | BLOB | Text (o), Number (allow multi-values) (o), Time (allow multi-values) (o), Rich Text, Formula (o) |
| | composite | Text (single-value) (o), Rich Text |
| | Date from Notes Rich Text field | |
| | number list | Number (o), Text (single-value) (o) |
| | datetime list | Time (o), Text (single-value) (o) |
| | text list | Text (o) |

## Create

| *LEI* | | *Notes* |
|---|---|---|
| Int | | Number (single-value) |
| Float | | Number (single-value) |
| Currency | | Number (single-value) |
| Numeric | | Number (single-value) |
| Datetime | | Time (single-value) |
| Text | bounded | Text (single-value) |
| | unbounded | Rich Text |
| Binary | BLOB | Text (single-value) |
| | composite | Rich Text |
| | number list | Number (allow multi-values) |
| | datetime list | Time (allow multi-values) |
| | text list | Text (allow multi-values) |

## Virtual Fields

The following virtual fields are supported by the Notes Connector. For more information, see the "Field Virtual Codes in the LCField Class" section of the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide*.

| *LEI* | | *Notes* |
|---|---|---|
| Int | | Number (single-value) |
| Float | | Number (single-value) |
| Currency | | Number (single-value) |
| Numeric | | Number (single-value) |
| Datetime | | Time (single-value) |
| Text | bounded | Text (single-value) |
| | unbounded | Rich Text |
| Binary | BLOB | Text (single-value) |
| | composite | Rich Text |
| | number list | Number (allow multi-values) |
| | datetime list | Time (allow multi-values) |
| | text list | Text (allow multi-values) |

# Chapter 7
# DB2 Connector

This chapter provides information about the DB2 Connector.

## Introduction to the DB2 Connector

The DB2 Connector allows you to define a Connection to DB2 via DB2 Connect Personal Edition, DB2 Connect Enterprise Edition or CAE (Client Application Enabler) for use with LEI. You can also connect to DB2 on an AS/400 or mainframe using a DDCS gateway. Refer to the *LEI Domino Connectivity and Installation Guide* for more information.

If you are running LEI native on the AS/400, you do not need to install additional DB2 software on the AS/400. DB2/400 is part of the base operating system and has support to be both a requester and server of remote DB2 requests. The sole requirement is that all DB2 databases being connected to, including the local DB2/400 database, be registered in the Relational Database Directory (see WRKRDBDIRE).

After creating your DB2 Connections, you should test each one of them using LCTEST, which is described in Chapter 5, "Introduction to Connectors".

## Terminology

The table below lists LEI terms and the corresponding terms in DB2.

| *Lotus Enterprise Integrator* | *DB2* |
|---|---|
| Server | N/A |
| Database | Database |
| Metadata | Table |
| Index | Index |
| Field | Column |
| Record | Row |

## Supported Characteristics

The following describes supported characteristics of the DB2 Connector.

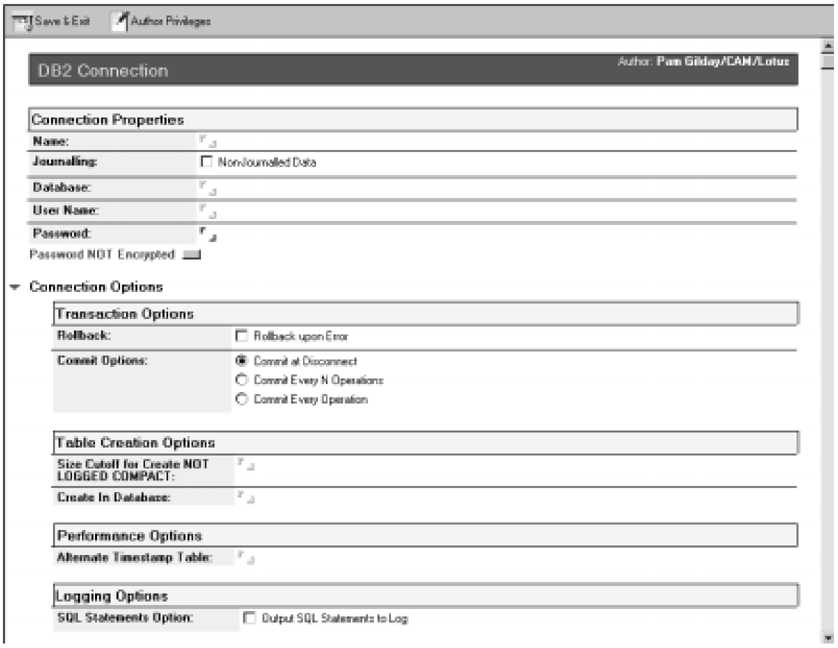
For a definition of the terms used in this section, see "Supported Connector Characteristics – List of Terms" in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for more information.

- Writeback support – Full except when the result set is ordered. In this case, writeback functionality is not available and is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
- Writeback index – Index strongly advised, but not required. Use the command "create index <indexname> on <tablename> (<column1>, <column2>, …, <columnN>)". (Simulated by keyed operations when cursors are not supported.)
- Statement syntax – Full DB2 SQL Syntax, including UDFs (User Defined Functions)
- Condition syntax – Valid when placed in the following DB2 statement: SELECT … WHERE <condition> ORDER BY …
- Array transfer – Insert
- Actions supported – Clear, Reset, Truncate, Commit, Rollback
- Catalog types – Database (no connection required), Metadata, Field, Procedure, Parameter
- Create types – Metadata, Index
- Drop types – Metadata, Index

**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections "Common Connector Properties" and "LCConnection Properties" in Chapter 2 "LConnection Class" of that same manual.

## DB2 Connection Document

The DB2 Connection Document, shown below, defines the name and location of a DB2 database. Choose Create – Connection – DB2, or click Create Connection and select DB2 from the list to access the DB2 Connection Document.



DB2 Connection

Author: Pam Gilday/CAM/Lotus

Connection Properties

| | |
|---|---|
| Name: | |
| Journaling: | ☐ Non-Journalled Data |
| Database: | |
| User Name: | |
| Password: | |

Password NOT Encrypted

Connection Options

Transaction Options

| | |
|---|---|
| Rollback: | ☐ Rollback upon Error |
| Commit Options: | ● Commit at Disconnect<br>○ Commit Every N Operations<br>○ Commit Every Operation |

Table Creation Options

| | |
|---|---|
| Size Cutoff for Create NOT LOGGED COMPACT: | |
| Create In Database: | |

Performance Options

| | |
|---|---|
| Alternate Timestamp Table: | |

Logging Options

| | |
|---|---|
| SQL Statements Option: | ☐ Output SQL Statements to Log |

## Connection Properties

| *Field* | *Description* |
|---|---|
| Name | Enter a unique name that identifies this Connection. |
| Database | Enter the identification of the DB2 database. |
| User Name | Enter the user name required to log in to the DB2 database. |
| Password | Enter the password associated with the user name. |

*continued*

| *Field* | *Description* |
|---|---|
| Non-Journalled Data | Select this option to enable support for SQL queries against non-SQL data external to DB2.<br><br>When the target is a non-journalled data source (for example, AS/400 database without journalling), SQL queries are not supported unless this option is selected. This option sets the access mode to read only (transaction isolation level of uncommitted read), which enables SQL queries. The default access level is read-write (transaction isolation level committed read).<br><br>The requirement for this option is most common when connecting to DB2/400 tables. It surfaces as a "<table> in <library> not valid for operation" error message.<br><br>**Note** Use of this option will disable all data insert and modification operations through this Connection. |

## Connection Options

The Connection Options section of the DB2 Connection document provides options for tuning the performance and actions of the Connection. Connection options are Activity-specific; they are stored with the Activity.

### Transaction Options

The Transaction Options section contains selections that apply to operations that create, modify, or delete data. Each of the fields in this section is described in the table below.

| *Field* | *Description* |
|---|---|
| Rollback Upon Error | When this option is enabled, and an Activity terminates in an error state, all changes in the current DB2 Connection's transaction are rolled back at disconnect time. |
| Commit at Disconnect | Enable this option to commit changes to the database as the database is being disconnected at the end of the operation. This is the default commit option. |
| Commit Every N Operations | Commits changes periodically after a specified number of changes are made. This number is specified as the Commit Frequency below. |
| Commit Frequency | This is used with the option Commit Every N Actions to specify how many actions must occur before the changes are committed.<br><br>This option only appears when Commit Every N is selected. |
| Commit Every Operation | When enabled, this option causes immediate commits for each change to the database. Note that this option reduces performance. |

### Table Creation Options

The Table Creation Options provide selections for managing table creation.

| *Field* | *Description* |
|---|---|
| Size Cutoff for Create NOT LOGGED COMPACT | When LEI creates a DB2 table, all rows are created with logging (the default). To specify the data size past which columns are created without logging (DB2 syntax is "NOT LOGGED COMPACT"), specify a value here. Text or Binary columns of unlimited length are always created with this option. |
| Create In Database | Use to add "IN DATABASE <dbname>" to a CREATE TABLE query constructed for table creation. |

### Performance Options

| *Field* | *Description* |
|---|---|
| Alternate Timestamp Table | When querying for the current timestamp from DB2 for timestamp Replication Activities, if DB2 is on an AS/400 or an OS/390 system, the query is run against the metadata specified by the Activity. If the table is large, this query can be inefficient. You can enter an alternate table name, which is smaller in size, to query for the timestamp. The table can be of any format and should contain as few rows as possible. This approach can improve the efficiency and speed of the timestamp query. |

### Logging Options

The Logging Options provide a feature for including SQL statements in the Activity log.

| *Field* | *Description* |
|---|---|
| Output SQL Statements to Log | This option causes any SQL statements that are generated during the Activity to be included in the Activity log. |

## Calling DB2 Stored Procedures

Wherever procedures are allowed in LEI, a DB2 stored procedure may be used. This includes RealTime Notes Activities, as the destination of a Direct Transfer, or within an LSX script. The stored procedure being called must have its parameters set up correctly for the call.

Input values are provided to DB2 procedures as named parameters. This requires that the parameters in DB2 use the same names as the LEI fields being provided as input values, The inputs being provided include key values when being used in the context of a keyed operation (selection, update, or delete context), and data values when relevant (insert or delete context). The input value datatypes are selected as the closest match to the datatype in the LEI system. Therefore it is crucial that the stored procedure not assume a particular datatype for input parameters, but rather validate and potentially convert the input parameter to the required datatype.

Any output from a DB2 stored procedure must be returned through a DB2 cursor left open in the stored procedure. The result set in this cursor will become the result set produced by the stored procedure.

The following is an example of a DB2 stored procedure written in C for the DB2 CLI preprocessor. This is the format that would be required for the Open event of a RealTime Notes Activity assuming that the key field is called NUMBER1 and the data fields are called NUMBER2, TEXT1, and TEXT2. In this context, the key field is the input parameter, and the result set is expected to include the data fields followed by the key field. Note that the input parameter datatype is checked and handled for both integer and float types.

```
SQL_API_RC SQL_API_FN sf_proc (void *reserved1,

                               void *reserved2,

                               struct sqlda *inout_sqlda,

                               struct sqlca *ca)

{

  /* Declare a local SQLCA */

  EXEC SQL INCLUDE SQLCA;


  /* Declare Host Variables */

  EXEC SQL BEGIN DECLARE SECTION;

    char *select_stmt  = "SELECT NUMBER2, TEXT1, TEXT2,
NUMBER1 FROM TALBENAME WHERE NUMBER1 = ?";
```

```
    long int number1;
  EXEC SQL END DECLARE SECTION;

  if (inout_sqlda)
  {
    if ((inout_sqlda->sqlvar[0].sqltype == SQL_TYP_INTEGER) ||
        (inout_sqlda->sqlvar[0].sqltype == SQL_TYP_NINTEGER))
      number1 = *((long int *)
(inout_sqlda->sqlvar[0].sqldata));
    else if ((inout_sqlda->sqlvar[0].sqltype == SQL_TYP_FLOAT)
||
             (inout_sqlda->sqlvar[0].sqltype ==
SQL_TYP_NFLOAT))
      number1 = (long int) *((double *)
(inout_sqlda->sqlvar[0].sqldata));
    else
      number1 = 0;
    EXEC SQL WHENEVER SQLERROR GOTO err;
    EXEC SQL PREPARE stmt FROM :select_stmt;
    EXEC SQL DECLARE curs CURSOR FOR stmt;
    EXEC SQL OPEN curs using :number1;
  }

err:
  memcpy(ca, &sqlca, sizeof (struct sqlca));
  if (inout_sqlda)
    *(inout_sqlda->sqlvar[0].sqlind) = -128;

  return(SQLZ_DISCONNECT_PROC);
}
```

## DB2 Data Types

The following tables show the correspondences of DB2 to LEI data types. In DB2, the data type pairs DOUBLE and FLOAT and NUMERIC and DECIMAL are synonymous, and only the first of each type synonym (DOUBLE and NUMERIC) is indicated in these tables.

- (p) – Indicates that if Allow Precision Loss is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not, an error is generated.

For more information on the organization of these tables, see the section "Note on Connection Data Type Conversion Tables" in Chapter 5.

**Note** These conversion tables may have changed since this printing. For the latest information about IBM's DB2 data types, see the IBM website.

### Execute

| *DB2* | *Attribute* | *Lotus Enterprise Integrator* |
|---|---|---|
| SMALLINT | | Int |
| INTEGER | | Int |
| DOUBLE or FLOAT | | Float |
| NUMERIC or DECIMAL | prec-scale<=9, scale<=0 | Int |
| | prec-scale<=19, scale<=4 | Currency |
| | other | Numeric |
| DATE | | Datetime |
| TIME | | Datetime |
| TIMESTAMP | | Datetime |
| CHAR | (default) | Text |
| | 4 BIT DATA | Binary (BLOB format) |
| VARCHAR | (default) | Text |
| | 4 BIT DATA | Binary (BLOB format) |
| LONG VARCHAR | (default) | Text |
| | 4 BIT DATA | Binary (BLOB format) |
| CLOB | | Text |
| GRAPHIC | | Text |

*continued*

| *DB2* | *Attribute* | *Lotus Enterprise Integrator* |
|---|---|---|
| VARGRAPHIC | | Text |
| LONG VARGRAPHIC | | Text |
| DBCLOB | | Text |
| BLOB | | Binary |

## Fetch

| *LEI* | | *DB2* |
|---|---|---|
| Int | | INTEGER, SMALLINT, DOUBLE (p), NUMERIC (p) |
| Float | | INTEGER, SMALLINT, DOUBLE, NUMERIC (p) |
| Currency | | INTEGER, SMALLINT, DOUBLE (p), NUMERIC (p) |
| Numeric | | INTEGER (p), SMALLINT (p), DOUBLE (p), NUMERIC (p) |
| Datetime | | TIMESTAMP (p), DATE, TIME |
| Text (o) | | CHAR [4 BIT DATA], VARCHAR [4 BIT DATA], LONG VARCHAR [4 BIT DATA], CLOB, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, BLOB |
| Binary (o) | BLOB | CHAR [4 BIT DATA], VARCHAR [4 BIT DATA], LONG VARCHAR [4 BIT DATA], CLOB, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, BLOB |
| | non-BLOB | Invalid |

## Insert/Update

| *LEI* | *DB2* |
|---|---|
| Int | INTEGER, SMALLINT (p), DOUBLE, NUMERIC (p) |
| Float | INTEGER (p), SMALLINT (p), DOUBLE, NUMERIC (p) |
| Currency | INTEGER (p), SMALLINT (p), DOUBLE (p), NUMERIC (p) |
| Numeric | INTEGER (p), SMALLINT (p), DOUBLE (p), NUMERIC (p) |
| Datetime | TIMESTAMP, DATE (p), TIME (p) |
| Text | CHAR [4 BIT DATA] (o), VARCHAR [4 BIT DATA] (o), LONG VARCHAR [4 BIT DATA] (o), CLOB (o), GRAPHIC (o), VARGRAPHIC (o), LONG VARGRAPHIC (o), DBCLOB (o), BLOB (o) |

*continued*

| *LEI* | | *DB2* |
|---|---|---|
| Binary | any | CHAR [4 BIT DATA] (o), VARCHAR [4 BIT DATA] (o), LONG VARCHAR [4 BIT DATA] (o), CLOB (o), GRAPHIC (o), VARGRAPHIC (o), LONG VARGRAPHIC (o), DBCLOB (o), BLOB (o) |
| | number list | INTEGER (p), SMALLINT (p), DOUBLE, NUMERIC (p) |
| | datetime list | TIMESTAMP, DATE (p), TIME (p) |

## Create

| *LEI* | | *DB2* |
|---|---|---|
| Int | | INT, SMALLINT, or NUMERIC (prec, 0) |
| Float | | DOUBLE or NUMERIC (prec, scale) |
| Currency | | NUMERIC (19, 4) |
| Numeric | | NUMERIC (prec, scale) |
| Datetime | | TIMESTAMP, DATE, or TIME |
| Text | | CHAR (fixed,len<=254), VARCHAR (254<len<=4000), LONG VARCHAR (4000<len<=32700), or CLOB (length > 32700) |
| Binary | BLOB | CHAR 4 BIT DATA (fixed,len<=254), VARCHAR 4 BIT DATA (254<len<=4000), LONG VARCHAR 4 BIT DATA (4000<len<=32700), or BLOB (length > 32700) |
| | composite | CHAR (fixed,len<=254), VARCHAR (254<len<=4000), LONG VARCHAR (4000<len<=32700), or CLOB (length > 32700) |
| | number list | DOUBLE |
| | datetime list | TIMESTAMP |
| | text list | CHAR (fixed,len<=254), VARCHAR (254<len<=4000), LONG VARCHAR (4000<len<=32700), or CLOB (length > 32700) |

## Accessing DB2 on an AS/400 System using the DB2 Connector

You can access DB2 on an AS/400 system using the DB2 Connector. The procedure varies slightly depending on whether or not the DB2 database resides on the same AS/400 system as the Domino Server.

### DB2 Database on same AS/400 System as Domino Server

Use the following procedure to access a DB2 database that resides on the same AS/400 system as the Domino server.

1. Add a relational database entry in DB2/400. An example is shown below:

   ```
   ADDRDBDIRE RDB(LPAR1A8M) RMTLOCALNAME(*LOCAL)
   ```

2. Create a DB2 Connection document using the relational database name in the above statement for the database name entry.
3. Enter the user name and password for the AS/400.

   This user name and password should have authority to access the given database.
4. If the table on the AS/400 system is not journalled, select the "Non-Journalled Data" option in the Connection Properties section of the DB2 Connection document.

### DB2 Database on Different AS/400 System than Domino Server

Use the following procedure to access a DB2 database that resides on a different AS/400 system than the Domino server.

1. Add a relational database entry in DB2/400. An example is shown below:

   ```
   ADDRDBDIRE RDB(LPAR1A8M) RTMLOCNAME(x.x.x.x)
   ```

   You can verify that the QRWLSNN job in QSYSWRK is running by using the WRKACTJOB command.
2. Start the licensing job on the remote AS/400 using the following command:

   ```
   STRTCPSRV *DDM.
   ```

3. Create a DB2 Connection document using the relational database name in the above statement for the database name entry.
4. Enter the user name and password for the AS/400.

   This user name and password should have authority to access the given database.

5. If the table on the AS/400 system is not journalled, select the "Non-Journalled Data" option in the Connection Properties section of the DB2 Connection document.

6. To use interactive SQL to verify that the Connection to the remote AS/400 is set up correctly, run STRSQL from your local AS/400 system. You must have 5769-ST1 installed to use STRSQL.

   **Note** If this is the first time that you have used SQL to connect to a remote AS/400 system, an error message (SQLPKG not found) may appear. You can run the following agent to create *SQLPKG on the remote AS/400 system:

   ```
   Option Public

   Uselsx "*lsxodbc"

   Sub Initialize

     Dim con As New ODBCConnection

     con.AutoCommit=False

     If (con.ConnectTo("rchasC9A","userid", "pwd")) Then

          Call con.Disconnect

     End If

     Print "Test Create SQL Pkg"

   End Sub
   ```

# Chapter 8
# EDA/SQL Connector

This chapter provides information about the EDA/SQL Connector.

## Introduction to the EDA/SQL Connector

The Lotus EDA/SQL Connector allows you to define a Connection to an EDA/SQL database.

EDA Server is a product offered by Information Builders, Inc. For information please contact Information Builders, Inc. at http://www.ibi.com. You must have the EDA/Client software for the operating system on which LEI is installed. You must also have an EDA Server on the platform where the EDA supported database resides. For complete details, see the *LEI Domino Connectivity and Installation Guide*.

After creating your EDA/SQL Connections, you should test each one of them using LCTEST, which is described in Chapter 5, "Introduction to Connectors." You can also find additional introductory information about Connectors in Chapter 5.

## Terminology

The table below lists LEI terms and the corresponding terms in EDA/SQL.

| *LEI* | *EDA/SQL* |
|---|---|
| Server | EDA Server |
| Database | Engine |
| Metadata | Table |
| Index | Index |
| Field | Column |
| Record | Row |

## Supported Characteristics

This following list captures supported characteristics of the EDA/SQL Connector.

For a definition of the terms used in this section, see “Supported Connector Characteristics – List of Terms” in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Domino Connector LotusScript Extensions Guide* for more information.

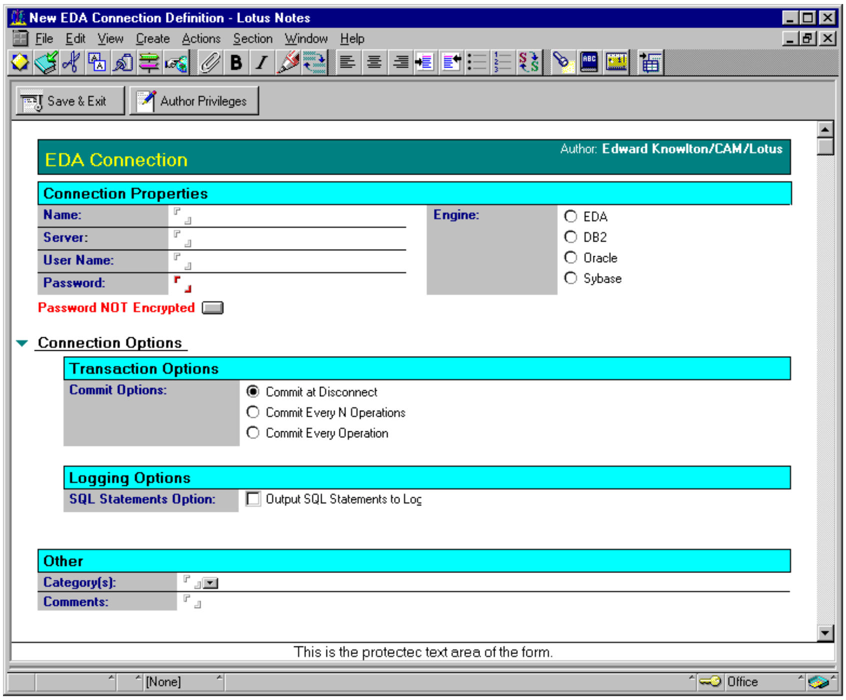
- Writeback support – N/A
- Writeback index – N/A
- Statement syntax – ANSI SQL syntax. To execute stored procedures, use the following syntax: “EXEC <stored procedure> [parm1] [,parm2] [,parm3] … [parmn]”
- Condition syntax – Valid when placed in the following EDA/SQL statement: SELECT … WHERE <condition> ORDER BY …
- Array transfer – NA
- Actions supported – Clear, Reset, Truncate, Commit, Rollback
- Catalog types – Server, Metadata, Field
- Create types – NA
- Drop types – NA

**Note** The EDA/SQL Connector does not support Update or Delete operations. Activities such as Replication and RealTime Notes will fail on any attempt to update or delete data.

**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections “Common Connector Properties” and “LCConnection Properties” in the LCConnection Class chapter of that same manual.

## EDA/SQL Connection Document

Use the EDA/SQL Connection Document to define a Connection to an EDA/SQL database. Select Create – Connection – EDA/SQL, or click Create Connection and select EDA/SQL, to access the EDA/SQL Connection Document.


New EDA Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

EDA Connection

Author: Edward Knowlton/CAM/Lotus

Connection Properties

Name:

Server:

User Name:

Password:

Engine: ○ EDA ○ DB2 ○ Oracle ○ Sybase

Password NOT Encrypted

Connection Options

Transaction Options

Commit Options: [● Commit at Disconnect] ○ Commit Every N Operations ○ Commit Every Operation

Logging Options

SQL Statements Option: ☐ Output SQL Statements to Log

Other

Category(s):

Comments:

This is the protectec text area of the form.

[None]

Office

## Connection Properties

The Connection Properties section of the document provides fields and options that establish the basic connectivity for this EDA/SQL Connection definition.

| *Field* | *Description* |
|---|---|
| Name | The name that identifies the Connection. |
| Database | EDA or the name of the relational database engine to use. Used for retrieval only, a database engine name (SYBASE, ORACLE or INFORMIX) is supplied when no EDA synonym exists or the relational database's native SQL syntax is needed. |

*continued*

| *Field* | *Description* |
|---|---|
| Server | Name of the EDA Server as listed in the EDA configuration. |
| User Name | The user name required to log in to the EDA Server. |
| Password | The password associated with the user name. |
| Engine | Select the database engine used to parse the SQL. |

**Note** When running on a UNIX platform, do not use EDA Connections as the Source and Destination links in the same Activity.

## Connection Options

There are two sets of options for the EDA/SQL Connector: Transaction Options and Logging Options. Connection Options are Activity-specific; they are stored with the Activity.

### Transaction Options

These advanced options let you specify the commit process for this Connection definition. These options apply to operations that insert data.

| *Field* | *Description* |
|---|---|
| Commit at Disconnect | Commits changes to the database as the database is being disconnected at the end of the operation. This is the default commit option. |
| Commit Every N Actions | Commits changes periodically after a specified number of changes are made. This number is specified as the Commit Frequency (see below). |
| Commit Frequency | Used with the option Commit Every N Actions to specify how many actions must occur before the changes are committed.<br>This option only appears when Commit Every N is selected. |
| Commit Every Action | Immediately commits each change to the database. This option reduces system performance because of the system overhead used to commit every action. |

### Logging Options

The Logging Options provide a feature for including SQL statements in the Activity log.

## EDA Data Types

The following tables show the correspondences of EDA/SQL to LEI data types for various types of Activity.

- (p) – Indicates that if Allow Precision Loss Only is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not, an error is generated.

For more information on the organization of these tables, see the section "Note on Connection Data Type Conversion Tables" in Chapter 5.

### Execute

| *EDA/SQL* | | *LEI* |
|---|---|---|
| Integer (I) | | Int |
| Single Float (F) | | Float |
| Double Float (D) | | Float |
| Decimal (P) | prec-scale<=9, scale<=0 | Int |
| | prec <= 15 | Float |
| | Other | Numeric |
| Alphanumeric (A) | | Text (fixed length, bound <= 254) |
| Date (YYMD) | | Datetime |

### Fetch

| *LEI* | *EDA/SQL* |
|---|---|
| *LEI* | Integer (I),Single Float (F), Double Float (D), Decimal (P), Alphanumeric (A) |
| *LEI* | Integer (I),Single Float (F), Double Float (D), Decimal (P), Alphanumeric (A) |
| *LEI* | Integer (I),Single Float (F), Double Float (D), Decimal (P), Alphanumeric (A) |
| *LEI* | Integer (I),Single Float (F), Double Float (D), Decimal (P), Alphanumeric (A) |
| *LEI* | Date (YYMD),Alphanumeric (A) |
| *LEI* | Alphanumeric (A) |
| *LEI* | Integer (I),Single Float (F), Double Float (D), Decimal (P), Date (YYMD),Alphanumeric (A) |

## Insert

| *LEI* | *EDA/SQL* |
|---|---|
| Int | Integer (I), Single Float (F), Double Float (D), Decimal (P), Alphanumeric (A) |
| Float | Integer (I),Single Float (F),Double Float (D),Decimal (P), Alphanumeric (A) |
| Currency | Integer (I), Single Float (F), Double Float (D), Decimal (P), Alphanumeric (A) |
| Numeric | Integer (I), Single Float (F), Double Float (D), Decimal (P),Alphanumeric (A) |
| Datetime | Date (YYMD), Alphanumeric (A) |
| Text | Integer (I), Single Float (F), Double Float (D), Decimal (P), Alphanumeric (A) |
| Binary | Alphanumeric (A) |

# Chapter 9
# File System Connector

This chapter provides information about the File System Connector.

## Introduction to the File System Connector

The File System Connector allows you to define a Connection to the file system on the server on which LEI is installed.

## Terminology

The table below lists LEI terms and the corresponding terms used by the File Connector.

| *LEI* | *File* |
|---|---|
| Server | N/A |
| Database | Directory path |
| Metadata | Subdirectory |
| Index | N/A |
| Field | N/A |
| Record | File specification |

## Supported Characteristics

The following list captures supported characteristics of the File System Connector.

For a definition of the terms used in this section, see “Supported Connector Characteristics – List of Terms” in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Domino Connector LotusScript Extensions Guide* for more information.

- Writeback support – Full
- Writeback index – N/A
- Statement syntax – File specification with optional wild cards
- Condition syntax – File specification with optional wild cards
- Array transfer – None
- Actions supported – Clear, Reset, Truncate
- Catalog types – Database, Metadata, Field
- Create types – Database, Metadata
- Drop types – Database, Metadata

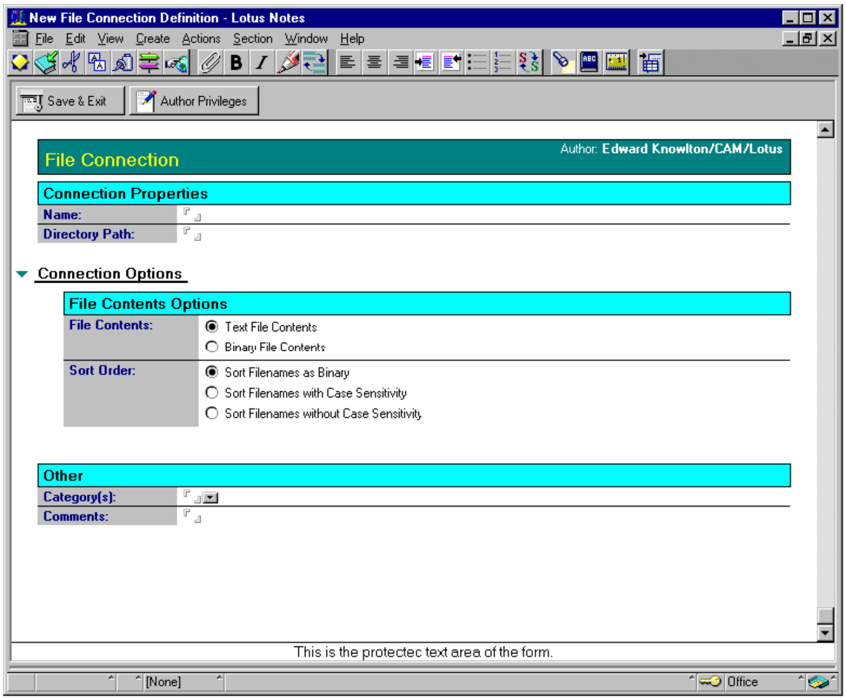
**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections “Common Connector Properties” and “LCConnection Properties” in the LCConnection Class chapter of that same manual.

**Note** Since the only key supported by the File Connection is the Filename field, Replication Activities must use this as the key field.

## File System Connection Document

The File System Connection document, shown below, defines a Connection to a file system. Choose Create – Connection – File to access the File System Connection document.

The fields in the File System Connection document are described in the table below. These fields are required.


New File Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

File Connection

Author: Edward Knowlton/CAM/Lotus

Connection Properties

Name:

Directory Path:

Connection Options

File Contents Options

File Contents: [✓ Text File Contents] Binary File Contents

Sort Order: [✓ Sort Filenames as Binary] Sort Filenames with Case Sensitivity Sort Filenames without Case Sensitivity

Other

Category(s):

Comments:

This is the protectec text area of the form.

## Connection Properties

The Connection Properties section of the document provides fields and options that establish the basic connectivity for this Connection definition.

| *Field* | *Description* |
|---|---|
| Connector Name | Enter a unique name to identify the Connector. |
| Directory Path | Enter the directory path containing the subdirectory to be used as metadata. |

## Connection Options

The File System Connection document has several sets of options. These options let you specify the type of contents in the files. Connection Options are Activity-specific; they are stored with the Activity.

### File Contents Options

Select the type of contents in the file.

| *Field* | *Description* |
|---|---|
| Text File Contents | Select this option to have file contents treated as text. |
| Binary File Contents | Select this option to have file contents treated as binary (BLOB) data. |

## Correspondence between the File System and the Connection and Activity Documents

The Directory Path field in the File System Connection document corresponds to the Database field in other Connection documents. The Subdirectory field in the Activity corresponds to the Table or Form name. In the Selection statement, you may use a file specification that includes wild cards such as “*.txt”.

The following example identifies all .txt files beginning with the letters “nab” in the directory path D:\files\somefiles\myfiles.

| *Field* | *Value* | *Note* |
|---|---|---|
| Directory Path (in Connection Document) | D:\files\somefiles | Equivalent to Database in other Connections |
| Subdirectory (in Activity document) | myfiles | Equivalent to Table or Form in other Connections |
| Select Statement | nab*.txt | |

- Timestamps on the FAT file system (older Win32) are precise to two seconds only and are always even. This may cause replication mismatches with more precise systems.
- The metadata returned from the File System Connector is Filename, Timestamp, Size, and Contents.
- Any Filename, Timestamp, or Size can be used as a key, and can be used in combination. Contents cannot be used as a key.

- Results can be ordered by Filename, Timestamp, or Size, although only one order field is permitted, resulting in only one key field for replication.
- The Select Metadata action button gives a list of subdirectories in the Directory path.

## File System Metadata

The metadata for a File record is significantly different from many other LEI Connections. There is a single record description (metadata) with set fields of set types. These fields are:

| | |
|---|---|
| Filename | This is an LEI TEXT field with a maximum length dependent on the maximum file path length (varies by operating system). The file name is specified without the directory, since that is defined by the DATABASE property. This field matches any LEI TEXT or BINARY field. Required for Insert operations as the file name to insert.<br>The filename field is returned as case-insensitive for Windows and NT operating systems. However, UNIX is case-sensitive. |
| Contents | This is an LEI TEXT (NATIVE format) or BINARY (BLOB format) field with no maximum length. This field contains the file contents as a stream, and matches any LEI TEXT or BINARY field. The default type is TEXT, but the BINARY type allows files to be handled as binary BLOBs. |
| ContentsFormat | This field is only available if you specify that the "Contents" field type is TEXT. The "ContentsFormat" property enables you to specify the format of the text as a specific character set. The character set specified is applied to the "Contents" field. Valid character set names are the same as the ones used for setting the "NativeText" property in the lei.ini file and the sorting character set in the Order MetaConnector. The default setting is "NATIVE." |
| Timestamp | This is an LEI DATETIME containing the last modification date and time for the file. This field maps to a LEI DATETIME field. |

*continued*

| | |
|---|---|
| *Size | This is an LEI INT field containing the size of the file contents, in bytes. This field is ignored on write operations; the size is obtained from the Contents field. This field matches any LEI number field (INT, FLOAT, CURRENCY, or NUMERIC), although any type other than INT will cause a precision loss error if Allow Precision Loss is disabled.<br>**Note** LEI removes trailing spaces. If Contents is type TEXT and its value includes trailing spaces, then there may be a difference between the value of the Size field and the length of the Contents field. |

**Note** Some databases and data sources have reserved words that cannot be used. Refer to the documentation for the product itself for information about reserved words.

For example, "Size" is an invalid column name in Oracle, so Activities involving a File System Connection and an Oracle Connection may encounter errors if "Create Metadata" is selected to create Oracle metadata with the same field names.

# Chapter 10
# Open Database Connectivity (ODBC) Connector

This chapter provides information about the ODBC Connector.

## Introduction to the ODBC Connector

Use an ODBC Connector to connect LEI to an ODBC-compliant database such as Lotus Approach. Lotus recommends using the Lotus ODBC drivers. See the Lotus ODBC Driver documentation for complete information.

**Note** LEI does not provide client software, you MUST have already installed client software. For example, if you are connecting to DB2 through ODBC, the system running LEI must have ODBC and the DB2 Client installed.

Refer to the *LEI Domino Connectivity and Installation Guide* for more information about connecting to an ODBC database.

## Terminology

The table below lists LEI terms and the corresponding terms in ODBC.

| *LEI* | *ODBC* |
|---|---|
| Server | Data Source |
| Database | Not Applicable |
| Metadata | Table |
| Index | Index |
| Field | Column |
| Record | Row |

## Supported Characteristics

The following list captures supported characteristics of the ODBC Connector.

For a definition of the terms used in this section, see “Supported Connector Characteristics – List of Terms” in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Domino Connector LotusScript Extensions Guide* for more information.

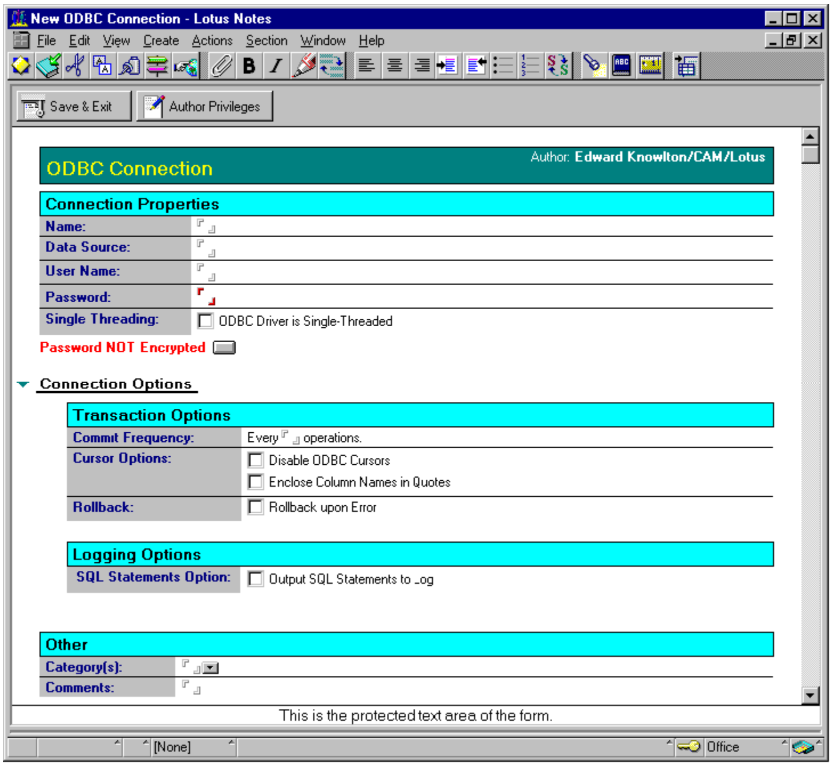
- Writeback support – Full except when the driver doesn’t support ordered updatable cursors. In this case, writeback functionality is not available and is simulated with keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
- Writeback index – Index strongly advised, but not required. Use the command “create index <indexname> on <tablename> (<column1>, <column2>, …, <columnN>)”, or use the native system syntax if this is not supported. (Simulated by keyed operations for drivers which do not support cursors).
- Statement syntax – Level 1 ODBC SQL Syntax
- Condition syntax – Valid when placed in the following ODBC statement: SELECT … WHERE <condition> ORDER BY …
- Array transfer – None
- Actions supported - Clear, Reset, Truncate, Commit, Rollback
- Catalog types – Server (no connection required) Metadata, Field, Procedure, Parameter
- Create types – Metadata
- Drop types – Metadata

**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections “Common Connector Properties” and “LCConnection Properties” in the LCConnection Class chapter of that same manual.

## ODBC Connection Document

The ODBC Connection document, shown below, defines a Connection to an ODBC database. Click the Create Connection icon in the Navigator and select ODBC to access the ODBC Connection document, or choose Create – Connection – ODBC.

The tables below describe the fields in the ODBC Connection document.


New ODBC Connection - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

ODBC Connection

Author: Edward Knowlton/CAM/Lotus

Connection Properties

| | |
|---|---|
| Name: | |
| Data Source: | |
| User Name: | |
| Password: | |
| Single Threading: | ☐ ODBC Driver is Single-Threaded |

Password NOT Encrypted

Connection Options

Transaction Options

| | |
|---|---|
| Commit Frequency: | Every operations. |
| Cursor Options: | ☐ Disable ODBC Cursors<br>☐ Enclose Column Names in Quotes |
| Rollback: | ☐ Rollback upon Error |

Logging Options

| | |
|---|---|
| SQL Statements Option: | ☐ Output SQL Statements to Log |

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

This is the protected text area of the form.

## Connection Properties

| *Field* | *Description* |
|---|---|
| Name | Enter a name to identify the Connection. |
| Data Source | Enter the name of the ODBC data source as listed in the ODBC Administrator. To indicate this property generically from the LC LSX or a MetaConnector, use the Server property. |
| User Name | Enter the user name required to log in to the ODBC data source. |
| Password | The password associated with the user name above. |
| Single Threading | Enable this option to notify the LEI Server that the ODBC driver being used is single-threaded, and that it does not support multi-threaded operation. |

## Connection Options

The ODBC Connection Options section enables you to specify options for the ODBC Connection. The fields in the ODBC Connection Options section are described below.

### Transaction Options

The options described in the following table apply to operations that create, modify, or delete data.

| *Field* | *Description* |
|---|---|
| Commit Frequency | Enter a number that specifies how many actions must occur before the changes are committed. |
| Disable ODBC Cursors | LEI automatically checks whether or not the data source supports ODBC cursors. Select this option to disable ODBC cursors and use keyed operations instead.<br>This may be required when the ODBC driver does not properly report support for ordered updateable cursors, causing an error for writeback operations (often from scripted Activities or replication). |
| Enclose Column Names in Quotes | When using column names with nonstandard characters (such as spaces, supported by some backends), select this option to enclose all column names in quotes. |
| Rollback Upon Error | Select this option if you want all changes in the current ODBC Connection's transaction rolled back at disconnect time when the Activity terminates in an error state. |

### Logging Options

| *Field* | *Description* |
|---|---|
| Output SQL Statements to Log | Select this option to include in the Activity log all SQL statements generated during processing of the Activity. |

## ODBC Data Types

The following tables show the correspondences of ODBC to LEI data types for various types of Activities.

- (p) – Indicates that if Allow Precision Loss Only is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not, an error is generated.

For more information on the organization of these tables, see the section "Notes on Connector Data Type Conversion Tables" in Chapter 5.

**Note** When using the IBM Client Access ODBC driver to insert data into the DB2/400 multi-byte data types ETYPE and JTYPE, DBCS data may need to be the exact length of the external type. When storing double-byte character set data to columns of these types, if the error "SQL0406 – Conversion error on assignment to column" is incorrectly generated, then the data being written should first be padded to the exact length of the DB2/400 column.

**Note** Only listed data types are supported.

## Execute

Note that since ODBC data sources may not support all ODBC data types in a table, the ODBC Connector will begin with the most likely type from the relevant row in the table below and continue until it finds a type supported in the current data source.

| *ODBC* | | *LEI* |
|---|---|---|
| BIT | | Int |
| TINYINT | | Int |
| SMALLINT | | Int |
| INTEGER | | Int |
| BIGINT | | Int |
| REAL | | Float |
| FLOAT | | Float |
| DOUBLE | | Float |
| DECIMAL | prec-scale<=9, scale<=0 | Int |
| DECIMAL | prec <= 15 | Float |
| DECIMAL | Other | Numeric |
| NUMERIC | prec-scale<=9, scale<=0 | Int |
| NUMERIC | prec <= 15 | Float |
| NUMERIC | other | Numeric |
| DATE | | Datetime |
| TIME | | Datetime |
| TIMESTAMP | | Datetime |
| CHAR | | Text (fixed length, bound <= 254) |
| VARCHAR | | Text (variable length, bound <= 254) |
| LONGVARCHAR | | Text (variable length, unbounded) |
| BINARY | | Binary (fixed length, bound <= 255) |
| VARBINARY | | Binary (variable length, bound <= 255) |
| LONGVARBINARY | | Binary (variable length, unbounded) |

## Fetch

Note that since ODBC data sources may not support all ODBC data types in a table, the ODBC Connector will begin with the most likely type from the relevant row in the table below and continue until it finds a type supported in the current data source.

| *LEI* | | *ODBC* |
|---|---|---|
| Int | | BIT, TINYINT, SMALLINT, INTEGER, BIGINT (o), REAL (o), FLOAT (p, o), DOUBLE (p, o), DECIMAL (p, o), NUMERIC (p, o) |
| Float | | BIT, TINYINT, SMALLINT, INTEGER, BIGINT (p), REAL, FLOAT, DOUBLE, DECIMAL (p, o), NUMERIC (p, o) |
| Currency | | BIT, TINYINT, SMALLINT, INTEGER, BIGINT (p, o), REAL (o), FLOAT (o), DOUBLE (o), DECIMAL (p, o), NUMERIC (p, o) |
| Numeric | | BIT, TINYINT (p, o), SMALLINT (p, o), INTEGER (p, o), BIGINT (p, o), REAL (p, o), FLOAT (p, o), DOUBLE (p, o), DECIMAL (p, o), NUMERIC (p, o) |
| Datetime | | DATE, TIME, TIMESTAMP |
| Text (o) | | CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONGVARBINARY |
| Binary (o) | BLOB | CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONGVARBINARY |
| | non-BLOB | Invalid |

## Insert/Update

Note that since ODBC data sources may not support all ODBC data types in a table, the ODBC Connector will begin with the most likely type from the relevant row in the table below and continue until it finds a type supported in the current data source.

| *LEI* | | *ODBC* |
|---|---|---|
| Int | | BIT (p), TINYINT (p, o), SMALLINT (p, o), INTEGER, BIGINT, REAL (p), FLOAT, DOUBLE, DECIMAL (p), NUMERIC (p) |
| Float | | BIT (p), TINYINT (p, o), SMALLINT (p, o), INTEGER(p, o), BIGINT (p, o), REAL (p, o), FLOAT, DOUBLE, DECIMAL (p), NUMERIC (p) |
| Currency | | BIT (p), TINYINT (p, o), SMALLINT (p, o), INTEGER(p, o), BIGINT (p, o), REAL (p), FLOAT (p, o), DOUBLE (p, o), DECIMAL (p), NUMERIC (p) |
| Numeric | | BIT, TINYINT (p, o), SMALLINT (p, o), INTEGER (p, o), BIGINT (p, o), REAL (p), FLOAT (p), DOUBLE (p), DECIMAL (p, o), NUMERIC (p, o) |
| Datetime | | DATE (p), TIME (o), TIMESTAMP. For Insert/Update into a TIME col, an overflow check is performed, not a precision check. |
| Text | | CHAR (o), VARCHAR (o), LONGVARCHAR, BINARY (o), VARBINARY (o), LONGVARBINARY |
| Binary | any | CHAR (o), VARCHAR (o), LONGVARCHAR, BINARY (o), VARBINARY (o), LONGVARBINARY |
| | number list | BIT (p), TINYINT (p, o), SMALLINT (p, o), INTEGER(p, o), BIGINT (p, o), REAL (p, o), FLOAT, DOUBLE, DECIMAL (p), NUMERIC (p) |
| | datetime list | DATE (p), TIME (p), TIMESTAMP |

## Create

Since ODBC data sources may not support all ODBC data types in a table, the ODBC Connector will begin with the most likely type from the relevant row in the table below and continue until it finds a type supported in the current data source.

| *LEI* | | *ODBC* |
|---|---|---|
| Int | | INTEGER, SMALLINT, TINYINT, BIGINT, DECIMAL, or NUMERIC |
| Float | | FLOAT, DOUBLE, REAL, DECIMAL, NUMERIC, or INTEGER |
| Currency | | NUMERIC, DECIMAL, FLOAT, DOUBLE, or INTEGER |
| Numeric | | NUMERIC, DECIMAL, FLOAT, DOUBLE, or INTEGER |
| Datetime | | TIMESTAMP, DATE, or TIME |
| Text | | CHAR (fixed, len<=254), VARCHAR (variable, len<=254), or LONGVARCHAR (len>254) |
| Binary | BLOB | BINARY(fixed, len<=255), VARBINARY (variable, len<=255), or LONGVARBINARY (len>255) |
| | composite | CHAR (fixed, len<=254), VARCHAR (variable, len<=254), or LONGVARCHAR (len>254) |
| | number list | FLOAT, DOUBLE, REAL, DECIMAL, NUMERIC, or INTEGER |
| | datetime list | TIMESTAMP, DATE, or TIME |
| | text list | CHAR (fixed, len<=254), VARCHAR (variable, len<=254), or LONGVARCHAR (len>254) |

# Chapter 11
# Oracle 7 Connector

This chapter provides information about the Oracle 7 Connector.

## Introduction to the Oracle 7 Connector

The Oracle 7 Connector allows you to define a Connection to an Oracle database.

Refer to the *LEI Domino Connectivity and Installation Guide* for information about connecting to an Oracle database.

Refer to Chapter 5, “Introduction to Connectors” for additional information about Connectors, including information on testing defined Connections.

## Terminology

The table below lists LEI terms and the corresponding terms in Oracle 7.

| *LEI* | *Oracle 7* |
|---|---|
| Server | Host String |
| Database | Not Applicable |
| Metadata | Table |
| Index | Index |
| Field | Column |
| Record | Row |

## Supported Characteristics

The following list captures supported characteristics of the Oracle 7 Connector.

For a definition of the terms used in this section, see "Supported Connector Characteristics – List of Terms" in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Domino Connector LotusScript Extensions Guide* for more information.

- Writeback support – Full
- Writeback index – Index strongly advised, but not required. Use the command "create index <indexname> on <tablename> (<column1>, <column2>, …, <columnN>)"
- Statement syntax – Full PL/SQL syntax. To execute stored procedures, use the following syntax: "BEGIN <stored procedure> (<parm1>, <parm2>, …, <parmN>) ; END ; ;". Note the extra semicolon required at the end of the statement.
- Condition syntax – Valid when placed in the following Oracle 7 statement: SELECT … WHERE <condition> ORDER BY …
- Array transfer – Fetch, Insert
- Actions supported – Clear, Reset, Truncate, Commit, Rollback
- Catalog types – Server (no connection required), Metadata, Index, Field, Procedure, Parameter
- Create types – Metadata, Index
- Drop types – Metadata, Index

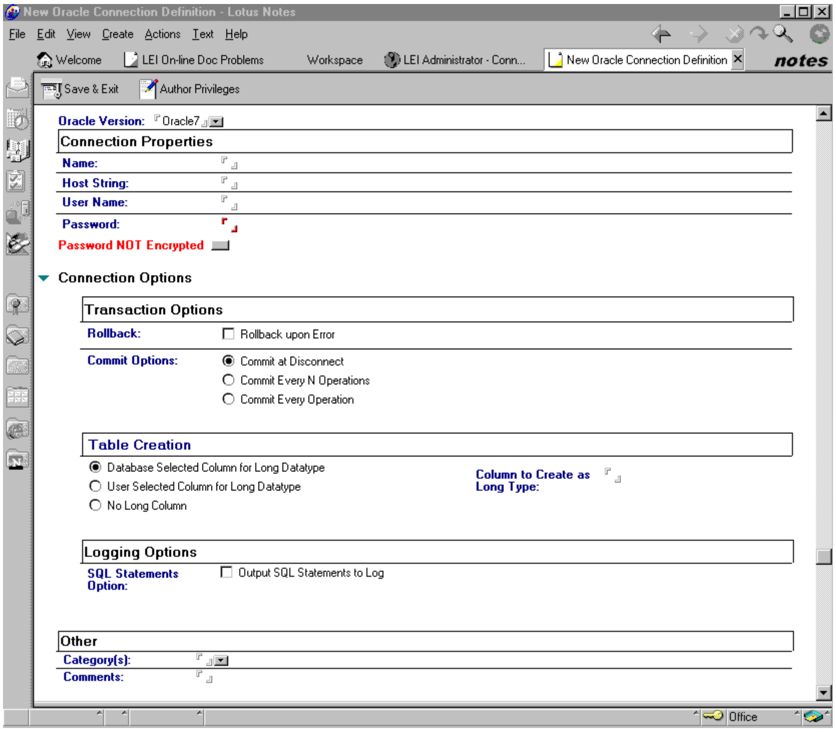
**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections "Common Connector Properties" and "LCConnection Properties" in the LCConnection Class chapter of that same manual.

In Oracle 7, a commit during a writeback operation and following an update unlocks the result set. Therefore, when a writeback result set is active, the Commit Frequency property (see Transaction Options section later in this chapter) is ignored and commits do not occur.

## Oracle Connection Document

The Oracle 7 Connection document, defines a connection to an Oracle 7 database.

Click the Create Connection icon from the Navigator and select Oracle 7, to access the Oracle 7 Connection document, or choose Create – Connection – Oracle from the menu.



Oracle Version: Oracle7

Connection Properties

Name:

Host String:

User Name:

Password:

Password NOT Encrypted

Connection Options

Transaction Options

Rollback: ☐ Rollback upon Error

Commit Options:
- [✓ Commit at Disconnect]
- Commit Every N Operations
- Commit Every Operation

Table Creation

- [✓ Database Selected Column for Long Datatype]
- User Selected Column for Long Datatype
- No Long Column

Column to Create as Long Type:

Logging Options

SQL Statements Option: ☐ Output SQL Statements to Log

Other

Category(s):

Comments:

## Connection Properties

Oracle 7 Connection properties are described in the following table.

| *Field* | *Description* |
|---|---|
| Connector Name | The name that identifies the Connector. |
| Host String | Identification of the Oracle 7 database server. The Host String can be for either SQL*Net V1 or SQL*Net V2 depending on what software you have configured for your Oracle 7 Server and LEI Server.<br>The general format for a V1 string is: network_prefix:server_name:sid.<br>The format for V2 consists of a single identifier: service_name. |
| User Name | The user name required to log in to the Oracle 7 database. |
| Password | The password associated with the user name above. |

## Connection Options

The Connection Options section provides options for managing data transactions, creating tables, and logging SQL commands.

The fields in the Oracle 7 Connection Options section are described below.

### Transaction Options

These options apply to operations that create, modify, or delete data.

| *Field* | *Description* |
|---|---|
| Rollback upon Error | When the Activity terminates in an error state, all changes in the current Oracle 7 Connection's transaction are rolled back at disconnect time. |
| Commit at Disconnect | Commits changes to the database as the database is being disconnected at the end of the operation. This is the default commit option. |
| Commit Every N Actions | Commits changes after a specified number of changes are made. This number is the value specified as the Commit Frequency. |
| Commit Every Action | Immediately commits each change to the database. This option reduces performance. |
| Commit Frequency | Used with the option Commit Every N Actions to specify how many actions must occur before the changes are committed. |

### Table Creation Options

In an Oracle 7 table there can only be one column of type LONG or LONGRAW. These options let you specify which column will be the long column, if any, during table creation. For more information, see the section entitled, “Oracle 7 Data Types.”

### Logging Options

The Logging Options provide a feature for including any SQL statements generated during an Activity to be included in the Activity Log.

| *Field* | *Description* |
|---|---|
| Output SQL Statements to Log | This option causes all SQL statements generated during the execution of the Activity to be included in the Activity’s log. This can be helpful when you need to troubleshoot the Activity. |

## The Oracle Connector and RealTime Activities

When you use data of type CHAR in a key field, the functionality of the Oracle Connector in two scenarios must be addressed: trimming spaces and creating an entry.

### Trim Spaces and the CHAR Data Type

When you use the Oracle Connector in a DECS RealTime Activity or in an LEI Notes RealTime Activity and you have a key field of data type of CHAR, do not select the “Trim spaces on all fields” option in the RealTime Activity document. The CHAR data type in Oracle requires a fixed number of spaces for it to be identified by the backend. Without the correct number of spaces, connection with the backend will be lost. If you must use a CHAR data type in a key field, Lotus recommends that you use the default setting, “Trim spaces on non-key fields.”

### Create Event and the CHAR Data Type

When you use data of type CHAR in a key field, certain events (such as updating and deleting) may not be available after a CREATE event has been executed. When Oracle creates a field it adds space padding to give the field a fixed length. However, Notes does not add additional space padding to the field during the CREATE event. This prevents Notes from locating the record on the Oracle side.

As a result, if fixed length text fields aren’t necessary, Lotus suggests using VARCHAR2 data types instead of CHAR data types. If you must use text fields of fixed length, use a formula filter to pad the specific field to the correct length. A code example for padding a field to the correct length is given below. This code can be used on the Notes form at the specific field under "Input Translation," in a DECS RealTime Activity document under "Options: When intercepting the creation of a document…", or in an LEI RealTime Notes Activity document under "Intercept Document Creation."

```
FIELD fieldName := fieldName;
```

```
@SetField ("fieldName"; fieldName + @Repeat (" ";
Oracle_Column_Length – @Length(fieldName)))
```

For example, if the Notes field name is "CharField" and the length of the Oracle CHAR column is 32, the formula would read:

```
FIELD CharField := CharField;
```

```
@SetField ("CharField"; CharField + @Repeat (" "; 32 –
@Length(CharField)))
```

When monitoring Document Create Events, the following options are available:

| *Option* | *Description* |
|---|---|
| Pre-Create Formula | A Notes formula language statement to execute on the new Notes document prior to the creation of a new record in the external database. For example: `FIELD LASTNAME:=@if(LASTNAME = "";"NA";LASTNAME);""` <br><br> **Note** If you are connecting to an Oracle backend and you must use text fields of fixed length, use a formula filter to pad the specific field to the correct length. See the Oracle 7 or 8 Connector documentation for more information. |
| StoredProcedure | This option executes a stored procedure in the external data source to store data that has been entered in the document. The RealTime key(s) and field(s) are supplied to the stored procedure as input parameters. <br><br> To see the fields that will be passed to the stored procedure, type the stored procedure name in this field and then press F9. The fields that will be passed to the stored procedure are displayed next to the stored procedure name. <br><br> **Note** LEI supports stored procedure browsing. |

## Calling Oracle 7 Stored Procedures

Wherever procedures are allowed in LEI, an Oracle 7 stored procedure may be used. This includes RealTime Notes Activities, as the destination of a Direct Transfer, or within an LEI LSX script. The stored procedure being called must have its parameters set up correctly for the call.

Input values are provided to Oracle 7 procedures as named parameters. This requires that the parameters in Oracle 7 use the same names as the LEI fields being provided as input values, The inputs being provided include key values when being used in the context of a keyed operation (selection, update, or delete context), and data values when relevant (insert or delete context). The input value datatypes provided by the Oracle 7 Connector are selected as the closest match to the datatype in the LEI system, and will be converted by Oracle 7 to the procedure parameter defined datatypes, as long as the conversion is supported by Oracle 7.

Oracle 7 differs from other RDBMS Connectors in that there is no way to return a result set from a procedure. Therefore, the Oracle 7 Connector supports output parameters as a way of returning results from a stored procedure. This requires additional information to be available at the time the procedure is called, specifically the context of the call and the output parameter names. This information will be automatically provided by LEI Activities, but must be manually specified when calling Oracle 7 procedures from an LEI LSX script. The context indicates whether the procedure should expect and specify output parameters, and the parameter names are provided as a property of the procedure call request to the Oracle 7 Connector. The output parameters must be standard datatypes – row sets may not be returned. This restricts the result set from an Oracle 7 procedure to a single row. Any parameters which are indicated as an input parameter and also in the output parameter list will be provided as input/output parameters.

The following is an example of an Oracle 7 stored procedure body. This is the format that would be required for the Open event of a RealTime Notes Activity assuming that the key field is called NUMBER1 and the data fields are called NUMBER2, TEXT1, and TEXT2. In this context, the key field is the input parameter, and the result set is expected to include the data fields and the key field. To accommodate the fact that one of the output parameter names is the same as a key value in the select statement, the parameter keys should be copied to local variables to avoid scoping problems in the procedure.

```
(NUMBER2 out tablename.number2%TYPE,

TEXT1 out tablename.text1%TYPE,
TEXT2 out tablename.text2%TYPE,
NUMBER1 in out tablename.number1%TYPE)
IS
BEGIN
  DECLARE number1_request tablename.number1%TYPE := number1;
  BEGIN
  SELECT t.NUMBER2, t.TEXT1, t.TEXT2, t.NUMBER1
    INTO number2, text1, text2, number1
    FROM tablename t
   WHERE t.NUMBER1 = number1_request;
  END;
END;
```

## Oracle 7 Data Types

The following tables show the correspondences of Oracle 7 to LEI data types for various types of Activities.

- (p) – Indicates that if Allow Precision Loss is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not an error is generated.

For more information on the organization of these tables, see the section "Notes on Connector Data Type Conversion Tables" in Chapter 5.

## Execute

| *Oracle 7* | | *LEI* |
|---|---|---|
| NUMBER | prec-scale<=9, scale<=0 | Int |
| | prec-scale<=15 | Float |
| | Other | Numeric |
| FLOAT | | Float |
| DATE | | Datetime |
| CHAR | | Text (fixed length, bound <= 255) |
| VARCHAR2 | | Text (variable length, bound <= 2000) |
| LONG | | Text (variable length, unbounded) |
| RAW | | Binary (variable length, bound <= 256 bytes) |
| LONG RAW | | Binary (variable length, unbounded) |
| ROWID | | Binary (BLOB format) (fixed length, bound <= 255) |
| MLSLABEL | | Binary (BLOB format) (fixed length, bound <= 255) |

**Note** When creating or accessing an LEI RealTime Activity and using the Oracle CHAR data type, do not use the RealTime Activity option "Trim spaces on all fields." The Oracle CHAR data type requires a fixed number of spaces in order to be correctly identified by the backend. Without these spaces, the connection to the backend will be lost. When using a CHAR data type as a key field, you should commonly use the default setting of "Trim spaces on non-key fields."

**Note** In an Oracle 7 table, there can only be one column of type LONG or LONGRAW. These options let you specify which column will be the long column, if any, during table creation.

Binary data cannot start in Oracle and go to Notes. Currently, the only field that can hold binary data is the rich text field, but it doesn't understand data that didn't originate from Notes. RAW and LONGRAW are binary data types. To use binary data types, the data must be inserted into the Oracle table from Notes first. This technique enables the data to obtain the necessary Notes formatting.

**Note** Because RAW and LONGRAW are binary data types, they must be specified in a rich text field in the source Notes database.

## Fetch

| *LEI* | | *Oracle 7* |
|---|---|---|
| Int | | FLOAT (p, o), NUMBER (p, o) |
| Float | | FLOAT (p), NUMBER (p) |
| Currency | | FLOAT (p, o), NUMBER (p, o) |
| Numeric | | FLOAT (p, o), NUMBER (p, o) |
| Datetime | | DATE |
| Text (o) | | CHAR, VARCHAR2, LONG, RAW, LONG RAW, ROWID, MLSLABEL |
| Binary (o) | BLOB | CHAR, VARCHAR2, LONG, RAW, LONG RAW, ROWID, MLSLABEL |
| | non-BLOB | Invalid |

## Insert/Update

| *LEI* | | *Oracle 7* |
|---|---|---|
| Int | | NUMBER, FLOAT |
| Float | | NUMBER (p), FLOAT (p) |
| Currency | | NUMBER (p), FLOAT (p) |
| Numeric | | NUMBER (p), FLOAT (p) |
| Datetime | | DATE (p) |
| Text | | CHAR (o), VARCHAR2 (o), LONG, RAW (o), LONG RAW (o), ROWID (o), MLSLABEL (o) |
| Binary | any | CHAR (o), VARCHAR2 (o), LONG, RAW (o), LONG RAW (o), ROWID (o), MLSLABEL (o) |
| | number list | NUMBER (p), FLOAT (p) |
| | datetime list | DATE (p) |

### Create

| *LEI* | | *Oracle 7* |
|---|---|---|
| Int | | NUMBER (prec,0) |
| Float | | FLOAT (prec) or NUMBER (prec, scale) |
| Currency | | NUMBER (19, 4) |
| Numeric | | NUMBER (prec, scale) |
| Datetime | | DATE |
| Text | | CHAR (fixed,len<=255), VARCHAR2 (255<len<=2000), or LONG (len>2000) |
| Binary | BLOB | RAW (fixed,len<=255) or LONG RAW (len>255) |
| | composite | CHAR (fixed,len<=255), VARCHAR2 (255<len<=2000), or LONG (len>2000) |
| | number list | FLOAT (prec) |
| | datetime list | DATE |
| | text list | CHAR (fixed,len<=255), VARCHAR2 (255<len<=2000), or LONG (len>2000) |

## Oracle 7 Field Mapping and NULL Entries

The Oracle 7 CHAR/VARCHAR value has three variations. They are string (length>0), empty string, and NULL. Other Oracle 7 types can have only two variations – value or NULL.

**Note** LEI regards a NULL as an undefined value, one that cannot be compared to other values. NULL is thus not equivalent to an empty string or to an empty value such as zero. For example, a replication will occur if LEI finds a NULL in one database matching an empty value in a replica. When this occurs, LEI uses SQL syntax to convert NULLS to values that are equivalent across databases.

### Transferring Data From Oracle 7 to Notes

The following statements pertain to Oracle 7 CHAR/VARCHAR data types and transferring data from Oracle 7 to Notes using LEI:

- If CHAR/VARCHAR contains some string (length > 0), the string will be transferred to Notes.
- If CHAR/VARCHAR field value is an empty string, LEI will create a Notes field with an empty value.
- If CHAR/VARCHAR field is NULL, LEI will not create any corresponding field in Notes.

The following statements pertain to Oracle 7 data types other than CHAR/VARCHAR and transferring data from Oracle 7 to Notes using LEI:

- If Oracle 7 field has value, the value will be transferred to the corresponding Notes field.
- If Oracle 7 field is NULL, LEI will not create any field in Notes.

### Transferring Data From Notes to Oracle 7

The following statements pertain to the Notes text field and transferring data from Notes to Oracle 7 using LEI:

- If the Notes text field contains a string (length > 0), the string will be transferred to Oracle 7.
- If the Notes text field is empty, an empty string will be inserted in the Oracle 7 field.
- If the Notes text field does not exist, NULL will be inserted in the Oracle 7 field.
- If the Notes field value exists, the value will be transferred to Oracle 7.
- If the Notes field exists but is empty, NULL will be inserted in Oracle 7.
- If the Notes field does not exist, NULL will be inserted in Oracle 7.

Since the Notes client user interface does not allow you to remove a field, LEI assumes empty Notes fields to be NULL.

## Transferring Data Between Oracle and Notes

The following table is helpful when transferring data between Oracle and Notes.

| *Oracle 7* | | *Notes* | |
|---|---|---|---|
| field type | field value | field type | field value |
| CHAR, VARCHAR | string (eg. 'ABC')<br>empty string ('')<br>NULL | TEXT | string (eg. 'ABC')<br>empty string, field exists<br>field does not exist |
| other types | value (eg. 12345)<br>NULL | corresponding type | value (eg. 12345)<br>field does not exist |

# Chapter 12
# Oracle 8 Connector

This chapter provides information about the Oracle 8 Connector.

## Introduction to the Oracle 8 Connector

The Oracle 8 Connector allows you to define a connection to an Oracle 8 database. The Oracle 8 Connector can be used with Lotus Enterprise Integration (LEI), Domino Enterprise Connection Services (DECS), Lotus Connector LotusScript Extensions (LC LSX and LEI LSX), and Lotus Connector Java Classes (LC Java).

Refer to the *Domino Connector Connectivity Guide* for information about connecting to an Oracle database, including information on testing defined Connections.

Refer to Chapter 5 “Introduction to Connectors” for additional information about Connectors.

## Terminology

The table below lists LEI terms and the corresponding terms in Oracle.

| *LEI* | *Oracle* |
|---|---|
| Server | Host String |
| Database | Not Applicable |
| Metadata | Table |
| Index | Index |
| Field | Column |
| Record | Row |
| Alt Metadata | View |

## Supported Characteristics

The following list captures supported characteristics of the Oracle 8 Connector.

For a definition of the terms used in this section, see “Supported Connector Characteristics – List of Terms” in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Domino Connector LotusScript Extensions Guide* for more information.

- Writeback support – Full writeback support. The Connection supports updatable cursors (update and remove)
- Writeback index – Index strongly advised, but not required. Use the command “create index <indexname> on <tablename> (<column1>, <column2>, …, <columnN>)”
- Statement syntax -- Full PL/SQL syntax. To execute stored procedures, use the following syntax: “BEGIN <stored procedure> (<parm1>, <parm2>, …, <parmN>) ; END ;”
- Condition syntax -- Valid when placed in the following Oracle statement: SELECT … WHERE <condition> ORDER BY …
- Actions supported – Clear, Reset, Truncate, Commit, Rollback
- Catalog types – Server (no connection required), Metadata, Index, Field, Procedure, Parameter
- Create types – Metadata, Index
- Drop types – Metadata, Index

**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections “Common Connector Properties” and “LCConnection Properties” in the LCConnection Class of that same manual.

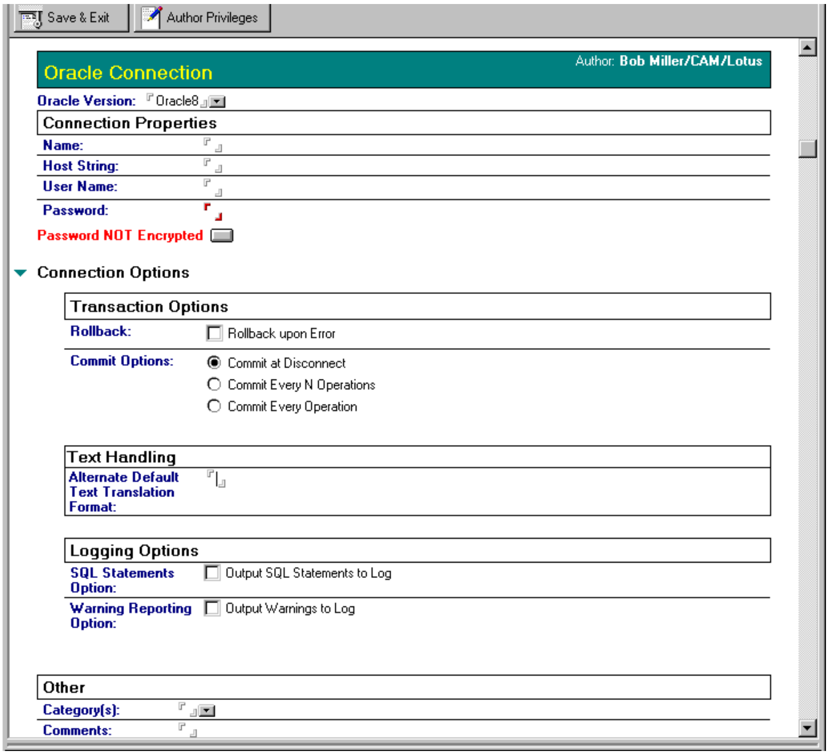
**Note** In Oracle 8, a commit during a cursored (writeback) operation following an update unlocks the result set. Therefore, when a writeback result set is active, the commit frequency property is ignored (commits do not occur).

## Oracle 8 Connection Document

The LEI Connection document for Oracle 8 defines a Connection to an Oracle database. Both of these documents are shown below.

To create a Connection document, select Create – Connector – ORACLE 8, or click Create Connection and select ORACLE 8, to access the ORACLE 8 Connection document. Alternatively, you can click the Create Connection icon from the Navigator and select Oracle.

The Oracle 8 Connection document for LEI is shown below.



Oracle Connection

Author: Bob Miller/CAM/Lotus

Oracle Version: Oracle8

Connection Properties

Name:

Host String:

User Name:

Password:

Password NOT Encrypted

Connection Options

Transaction Options

Rollback: ☐ Rollback upon Error

Commit Options: [✓ Commit at Disconnect] Commit Every N Operations / Commit Every Operation

Text Handling

Alternate Default Text Translation Format:

Logging Options

SQL Statements Option: ☐ Output SQL Statements to Log

Warning Reporting Option: ☐ Output Warnings to Log

Other

Category(s):

Comments:

## Fields in the Connection Document

See the following tables for descriptions of the Connection document fields.

### Oracle Version

| *Field* | *Description* |
|---|---|
| Oracle Version | Select the version of the Oracle Client that you want to use with this Connector.<br>**Caution** If you have an existing Oracle 7 Connection document and you change the "Oracle Version" to Oracle 8 you may get inconsistent and unexpected results. |

### Connection Properties

| *Field* | *Description* |
|---|---|
| Name | The name that you use to identify the Connector. (LEI only) |
| Host String | Service Name as defined in your local tnsnames.ora file. Service names are simple names commonly used to specify the location of database services on a network. Examples of Service names are: "flood" or "flood.world." |
| User Name | The user name required to log in to the Oracle database. |
| Password | The password associated with the user name above. |

### Connection Options

The Connection Options section provides options for managing data transactions, creating tables, and logging SQL commands.

The fields in the Oracle Connection Options section are described below.

### Transaction Options

These options apply to operations that create, modify, or delete data.

| *Field* | *Description* |
|---|---|
| Rollback upon Error | When the Activity terminates in an error state, all changes in the current Oracle Connection's transaction are rolled back at disconnect time. |
| Commit at Disconnect | Commits changes to the database as the database is being disconnected at the end of the operation. This is the default commit option. |
| Commit Every N Actions | Commits changes periodically after a specified number of changes are made. This number is the value specified as the Commit Frequency. |

*continued*

| *Field* | *Description* |
|---|---|
| Commit Every Action | Immediately commits each change to the database. This option reduces performance. |
| Commit Frequency | Visible only when “Commit Every N Actions” is selected. Use “Commit Frequency” to specify how many actions must occur before the changes are committed. |

## Text Handling

Text handling in Oracle 8 makes it possible to override the existing text translation format with an alternative default.

| *Field* | *Description* |
|---|---|
| Alternative Default Text Translation Format | Use this option to enter an alternative default Oracle text format if the standard default results in improper translation. The text format must be supported by Oracle. |

## Logging Options

The Logging Options provide a feature for including any SQL statements generated during an Activity to be included in the Activity log.

| *Field* | *Description* |
|---|---|
| SQL Statements Option: Output SQL Statements to Log | This option causes all SQL statements generated during the execution of the Activity to be included in the Activity’s log. This can be helpful when you need to troubleshoot the Activity. |
| Warning Reporting Option | This option causes warnings generated during the execution of the Activity to be included in the Activity’s log. |

## The Oracle 8 Connector and RealTime Activities

When you use the CHAR datatype in a key field, Oracle Connector functionality must be addressed with regard to the following two scenarios:

- Trimming spaces
- Creating an event entry

### Trimming Spaces and the CHAR Data Type

When you use the Oracle Connector in an LEI Notes RealTime Activity and you have a key field of data type CHAR, do not select the “Trim spaces on all fields” option in the RealTime Activity document. The CHAR data type in Oracle requires a fixed number of spaces for it to be identified by the backend. Without the correct number of spaces, Connection with the backend will be lost. If you must use a CHAR data type in a key field, Lotus recommends that you use the default setting, “Trim spaces on non-key fields.”

### Creating and Event and the CHAR Data Type

When you use data of type CHAR in a key field, certain events (such as updating and deleting) may not be available after a CREATE event has been executed. When Oracle creates a field it adds space padding to give the field a fixed length. However, Notes does not add additional space padding to the field during the CREATE event. This prevents Notes from locating the record on the Oracle side.

As a result, if fixed length text fields aren’t necessary, Lotus suggests using VARCHAR2 data types instead of CHAR data types. If you must use text fields of fixed length, use a formula filter to pad the specific field to the correct length. A code example for padding a field to the correct length is given below. This code can be used on the Notes form at the specific field under “Input Translation,” in an LEI RealTime Activity document under “Options: When intercepting the creation of a document…”, or in an LEI RealTime Notes Activity document under “Intercept Document Creation.”

```
FIELD fieldName := fieldName;

@SetField ("fieldName"; fieldName + @Repeat (" ";
Oracle_Column_Length –@Length(fieldName)))
```

For example, if the Notes field name is “CharField” and the length of the Oracle CHAR column is 32, the formula would read as follows:

```
FIELD CharField := CharField;

@SetField ("CharField"; CharField + @Repeat (" "; 32
–@Length(CharField)))
```

## Calling Oracle Stored Procedures

Wherever procedures are allowed in LEI, an Oracle stored procedure may be used. This includes RealTime Notes Activities, as the destination of a Direct Transfer, or within an LC LSX script. The stored procedure being called must have its parameters set up correctly for the call.

Input values are provided to Oracle procedures as named parameters. This requires that the parameters in Oracle use the same names as the LEI fields being provided as input values, The inputs being provided include key values when being used in the context of a keyed operation (selection, update, or delete context), and data values when relevant (insert or delete context). The input value datatypes provided by the Oracle Connector are selected as the closest match to the datatype in the LEI system, and will be converted by Oracle to the procedure parameter defined datatypes, as long as the conversion is supported by Oracle.

The Oracle Connector supports output parameters as the way of returning results from a stored procedure. This requires additional information to be available at the time the procedure is called, specifically the context of the call and the output parameter names. This information will be automatically provided by LEI Activities, but must be manually specified when calling Oracle procedures from an LSX script. The context indicates whether the procedure should expect and specify output parameters, and the parameter names are provided as a property of the procedure call request to the Oracle Connector. The output parameters must be standard datatypes; row sets may not be returned. This restricts the result set from an Oracle procedure to a single row. Any parameters which are indicated as an input parameter and also in the output parameter list will be provided as input/output parameters.

**Note** Oracle 8 limits LONG data types to 32k for stored procedures.

**Note** The Connector does not support LOB data types for stored procedures.

## Example

The following is an example of an Oracle stored procedure body. This is the format that would be required for the Open event of a RealTime Notes Activity assuming that the key field is called NUMBER1 and the data fields are called NUMBER2, TEXT1, and TEXT2. In this context, the key field is the input parameter, and the result set is expected to include the data fields and the key field. To accommodate the fact that one of the output parameter names is the same as a key value in the select statement, the parameter keys should be copied to local variables to avoid scoping problems in the procedure.

```
(NUMBER2 out tablename.number2%TYPE,

TEXT1 out tablename.text1%TYPE,

TEXT2 out tablename.text2%TYPE,

NUMBER1 in out tablename.number1%TYPE)

IS

BEGIN

  DECLARE number1_request tablename.number1%TYPE := number1;

  BEGIN

  SELECT t.NUMBER2, t.TEXT1, t.TEXT2, t.NUMBER1

    INTO number2, text1, text2, number1

    FROM tablename t

   WHERE t.NUMBER1 = number1_request;

  END;

END;
```

## Oracle 8 Data Types

The following tables show the correspondences of Oracle to LEI data types for various types of Activities.

- (p) – Indicates that if Allow Precision Loss is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not an error is generated.

**Caution** You can specify only one LONG for each column/statement.

**Caution** If you want to use BLOBs or RAWs they must originate in Notes, then be inserted into the Oracle table, and then be read back into Notes, otherwise they will not be in a format that Notes can recognize. This is true for LEI when a Notes database is one of the connected databases.

**Caution** In Oracle 8.0.5 and earlier, CLOBS do not support multi-byte varying-width character sets. However, Oracle 8i does support multi-byte varying-width character sets. To avoid this problem in Oracle 8.0.5 and earlier, use LONG or VARCHAR2 data types instead of CLOB.

### Execute

| *Oracle* | | *Lotus Connector API (used by LEI)* |
|---|---|---|
| NUMBER | prec-scale<=9, scale<=0 | Int |
| | prec-scale<=15 | Float |
| | Other | Numeric |
| FLOAT | | Float |
| DATE | | Datetime |
| CHAR | | Text (fixed length, bound <= 2000) |
| VARCHAR2 | | Text (variable length, bound <= 4000) |
| LONG | | Text (variable length, unbounded) |
| CLOB | | Text (variable length, unbounded) |
| RAW | | Binary (BLOB format) (variable length, bound <= 2000) |
| LONG RAW | | Binary (BLOB format) -- variable length, unbounded |
| BLOB | | Binary (BLOB format) -- variable length, unbounded |

## Fetch

| *Oracle* | *Lotus Connector API (used by LEI)* | |
|---|---|---|
| FLOAT (p, o), NUMBER (p, o) | Int | |
| FLOAT (p), NUMBER (p) | Float | |
| FLOAT (p, o), NUMBER (p, o) | Currency | |
| FLOAT (p, o), NUMBER (p, o) | Numeric | |
| DATE | Datetime | |
| CHAR, VARCHAR2, LONG, RAW, LONG RAW, CLOB, BLOB | Text (o) | |
| CHAR, VARCHAR2, LONG, RAW, LONG RAW, CLOB, BLOB | Binary (o) | BLOB/composite |

## Insert/Update

| *Lotus Connector API (used by LEI)* | | *Oracle* |
|---|---|---|
| Int | | NUMBER, FLOAT |
| Float | | NUMBER (p), FLOAT (p) |
| Currency | | NUMBER (p), FLOAT (p) |
| Numeric | | NUMBER (p), FLOAT (p) |
| Datetime | | DATE (p) |
| Text | | CHAR (o), VARCHAR2 (o), LONG, RAW (o), LONG RAW, ROWID (o), CLOB, BLOB |
| Binary | any | CHAR (o), VARCHAR2 (o), LONG, RAW (o), LONG RAW, ROWID (o), CLOB, BLOB |
| | number list | NUMBER (p), FLOAT (p) |
| | datetime list | DATE (p) |

## Create

| *Lotus Connector API (used by LEI)* | | *Oracle* |
|---|---|---|
| Int | | NUMBER (prec,0) |
| Float | | FLOAT (prec) or NUMBER (prec, scale) |
| Currency | | NUMBER (19, 4) |
| Numeric | | NUMBER (prec, scale) |
| Datetime | | DATE |
| Text | | CHAR (fixed, len<=2000), VARCHAR2 (2000<len<=4000), or CLOB (len>=4000) |
| Binary | BLOB | RAW (fixed, len<=2000) or BLOB (len>2000) |
| | composite | CHAR (fixed, len<=2000), VARCHAR2 (2000<len<=4000), or CLOB (len>4000) |
| | number list | FLOAT (prec) |
| | datetime list | DATE |
| | text list | CHAR (fixed, len<=2000), VARCHAR2 (2000<len<=4000), or CLOB (len>4000) |

## Oracle 8 Connector Properties

See Appendix B of the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of properties for the Oracle 8 Connector.

# Chapter 13
# OLE DB

This chapter provides information about the OLE DB Connector.

## Introduction to the OLE DB Connector

This chapter provides information about the Domino Connector for OLE DB. It contains information about how to create Microsoft SQL Server Connections.

Use the Domino Connector for OLE DB to connect LEI to a Microsoft SQL Server database. Since Microsoft only provides the OLE DB technology on Windows platforms, use the ODBC Connector to connect to SQL Server databases when LEI is installed on non-Windows platforms.

## Terminology

The table below lists LEI terms and the corresponding terms in OLE DB.

| *LEI* | *OLE DB* |
|---|---|
| Server | Provider |
| Database | Database/Catalog |
| Metadata | Table/View |
| Index | Index |
| Field | Field/Column |
| Record | Record/Row |

## Supported Characteristics

The following table captures supported characteristics of the OLE DB.

For a definition of the terms used in this section, see “Supported Connector Characteristics – List of Terms” in Chapter 5 of this manual. If you are accessing the Connector programmatically, see Appendix B and C in the *Domino Connector LotusScript Extensions Guide* for more information.

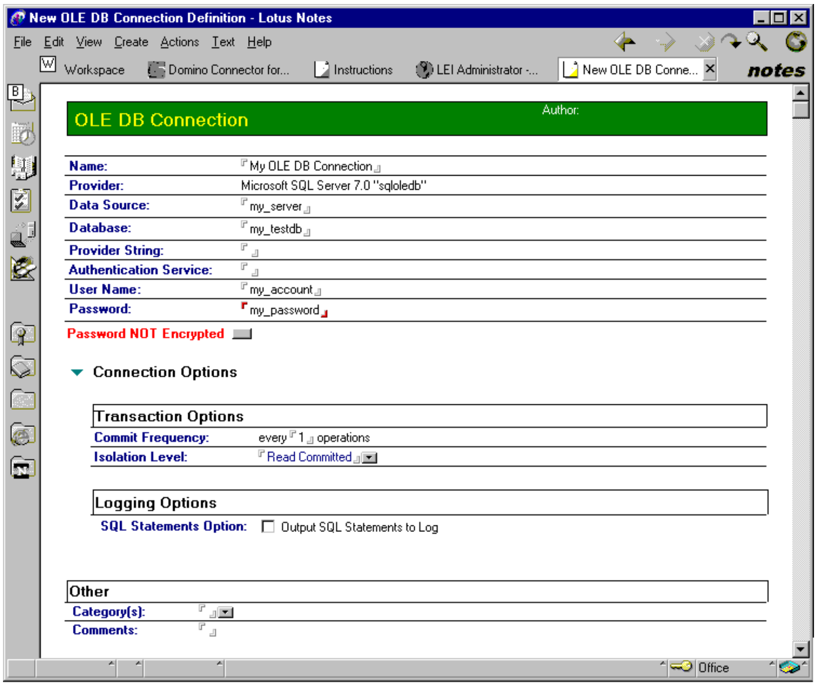
- Writeback support – Depending on the OLE DB provider, writeback support is not always available. If it is not available, it will be simulated by the OLE DB Connector using keyed operations. Make sure that the proper key settings and values are provided even when performing writeback update and remove operations.
- Writeback index – Index required for SQL Server. Use the command “create index <indexname> on <tablename> (<column1>, <column2>, …, <columnN>)”.
- Statement syntax – Dependent on the OLE DB Provider. For Microsoft SQL Server Full Transact-SQL syntax is supported, including stored procedures.
- Condition syntax – Valid when placed in the following SQL statement: SELECT … WHERE <condition> ORDER BY
- Array transfer – None
- Actions supported – Clear, Reset, Truncate, Commit, Rollback
- Catalog types – Database, Metadata, Index, Field, Procedure, Parameter
- Create types – Metadata, Index
- Drop types – Metadata, Index

  **Note** The Database Catalog type requires a valid Connection. If you attempt to use this data type without a valid Connection, you will receive an error message.

## OLE DB Connection Document

The LEI Connection document for OLE DB defines a connection to a SQL Server. This document is shown below.

To create a Connection document, select Create – Connector – OLE DB, or click Create Connection and select OLE DB.



OLE DB Connection

Author:

| | |
|---|---|
| Name: | My OLE DB Connection |
| Provider: | Microsoft SQL Server 7.0 "sqloledb" |
| Data Source: | my_server |
| Database: | my_testdb |
| Provider String: | |
| Authentication Service: | |
| User Name: | my_account |
| Password: | my_password |

Password NOT Encrypted

Connection Options

Transaction Options

| | |
|---|---|
| Commit Frequency: | every 1 operations |
| Isolation Level: | Read Committed |

Logging Options

SQL Statements Option: ☐ Output SQL Statements to Log

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

### Fields in the OLE DB Connection Document

See the following tables for descriptions of the LEI Connection document fields.

**Note** Refer to Chapter 5, "Introduction to Connectors" for additional information about Connectors and Connection documents in general. LSX users can refer to the LSX documentation.

## Connection Properties

| *Field* | *Description* |
|---|---|
| Name | Enter a unique name that identifies the Connector.<br>**Warning** LEI Activities and MetaConnectors access this Connection by its name. If you change this Connection's name then documents that use the Connection might not be able to find it. |
| Provider | For this release, this property is fixed as Microsoft SQL Server. |
| Data Source | Enter the network name of the server running SQL Server. |
| Database | Enter the name of the SQL Server database that you want to connect to. |
| Provider String | Enter any required provider string values. SQLOLEDB recognizes an ODBC-like syntax in provider string property values. Within the string, elements are delimited by using a semicolon. The final element in the string must be terminated with a semicolon. Each element consists of a keyword, an equal sign character, and the value passed on initialization, for example:<br>`Server=Gumby;UID=george;`<br>You may need to set 'network' to the name of the Net-Library (DLL) used to communicate with the SQL Server. The name should not include the path or the .dll file name extension. For example to use the TCP/IP network library, add the following:<br>`Network=DBMSSOCN;`<br>Alternatively, the default Net-Library can be set by the SQL Server Client Network Utility. |
| Authentication Service | If this property is set to "SSPI", SQLOLEDB uses Windows NT Authentication Mode to authorize user access to the SQL Server database.<br>If it is not set, SQL Server security is used. The SQL Server login and password are specified in the User Name and Password properties. |
| User Name | Enter the user name required to log in to the SQL Server. Leave blank if using Windows NT Authentication. |
| Password | Enter the password associated with the user name above. Leave blank if using Windows NT Authentication. |

## Transaction Options

| *Field* | *Description* |
|---|---|
| Commit Frequency | Indicates how often to commit a transaction. If set to 1, all actions are immediately committed. If set to 0 (zero), actions are committed at disconnect. Set this option to 1 for Replication Activities. |
| Isolation level | This property determines the extent that outside actions can affect a transaction. The options are as follows:<br>• Read Uncommitted — A transaction operating at the Read Uncommitted level can see uncommitted changes made by other transactions. At this level of isolation, dirty reads, non-repeatable reads, and phantoms are all possible.<br>• Read Committed — A transaction operating at the Read Committed level cannot see changes made by other transactions until those transactions are committed. At this level of isolation, dirty reads are not possible but non-repeatable reads and phantoms are possible<br>• Repeatable Read — A transaction operating at the Repeatable Read level is guaranteed not to see any changes made by other transactions in values it has already read. At this level of isolation, dirty reads and nonrepeatable reads are not possible but phantoms are possible.<br>• Serializable — A transaction operating at the Serializable level guarantees that all concurrent transactions interact only in ways that produce the same effect: as if each transaction were entirely executed one after the other. At this isolation level, dirty reads, nonrepeatable reads, and phantoms are not possible. |

## Logging Options

| *Field* | *Description* |
|---|---|
| SQL Statements Option: Output SQL Statements to Log | Select this option to include all SQL statements generated during the processing of the Activity in the Activity log.<br>**Note** Selecting this option slows down performance. Use it only for troubleshooting. |

## OLE DB Connector Operational Considerations

Below are some important considerations when using the OLE DB Connector with Microsoft SQL Server:

- A Microsoft SQL Server table that is the destination for a Replication Activity must have a unique index. Submit the following command through ISQL or another SQL front end to create the index before attempting the Replication:

  ```
  create unique index <indexname> on <tablename> (<col1>,
  <col2>,..., <colN>)
  ```

  **Note** You can use just one column or many columns, but for efficient replication, the replication keys should be included in the index. To remove an index, use the command:

  ```
  drop index <tablename>.<indexname>
  ```

- When doing Timestamp Replication using the LEI Replication Activity, a Microsoft SQL Server timestamp type column (an internal type) cannot be used as the timestamp. It must be a datetime or smalldatetime type.
- Binary is a fixed-length data type. Use this type when you expect the data entries in a column to be exactly the same size. SQL Server 7.0 appends zero bits to entries where the binary data is less than the column length. When this occurs, the data can no longer be read by Notes.

  For example, if you have a Notes rich text field mapped to a SQL Server 7.0 binary type in a RealTime Activity, any data entered into Notes will be saved in SQL Server 7.0. When you attempt to retrieve the data from SQL Server through Notes, through the RealTime Activity, the data will be corrupted. In this case, use the image type instead of binary. Image is capable of handling data that is greater than 8KB without truncation errors.

## Calling MS SQL Server Stored Procedures

Wherever procedures are allowed in LEI, you can use a Microsoft SQL Server stored procedure. This includes RealTime Notes Activities, as the destination of a Direct Transfer, or within an LSX script. The stored procedure being called must have its parameters set up correctly for the call.

Input values are provided to SQL procedures as named parameters. This means that the parameters in SQL must use the same names as the LEI fields being provided as input values. These inputs include key values when used in the context of a keyed operation (selection, update, or delete context), and data values when relevant (insert or delete context). The input value datatypes are selected as the closest match to the datatype in LEI, and are converted by SQL to the procedure parameter defined datatypes, as long as the conversion is supported by Microsoft SQL Server.

Any output from a stored procedure must be returned through a final SELECT statement in the procedure. The result set of this selection will become the result set produced by the stored procedure.

The following is an example of a Microsoft SQL Server stored procedure. This is the format that would be required for the Open event of a RealTime Notes Activity assuming that the key field is called NUMBER1 and the data fields are called NUMBER2, TEXT1, and TEXT2. In this context, the key field is the input parameter, and the result set is expected to include the data fields followed by the key field.

```
create procedure OpenProc @NUMBER1 int

as select NUMBER2, TEXT1, TEXT2, NUMBER1 from tablename

where NUMBER1 = @NUMBER1
```

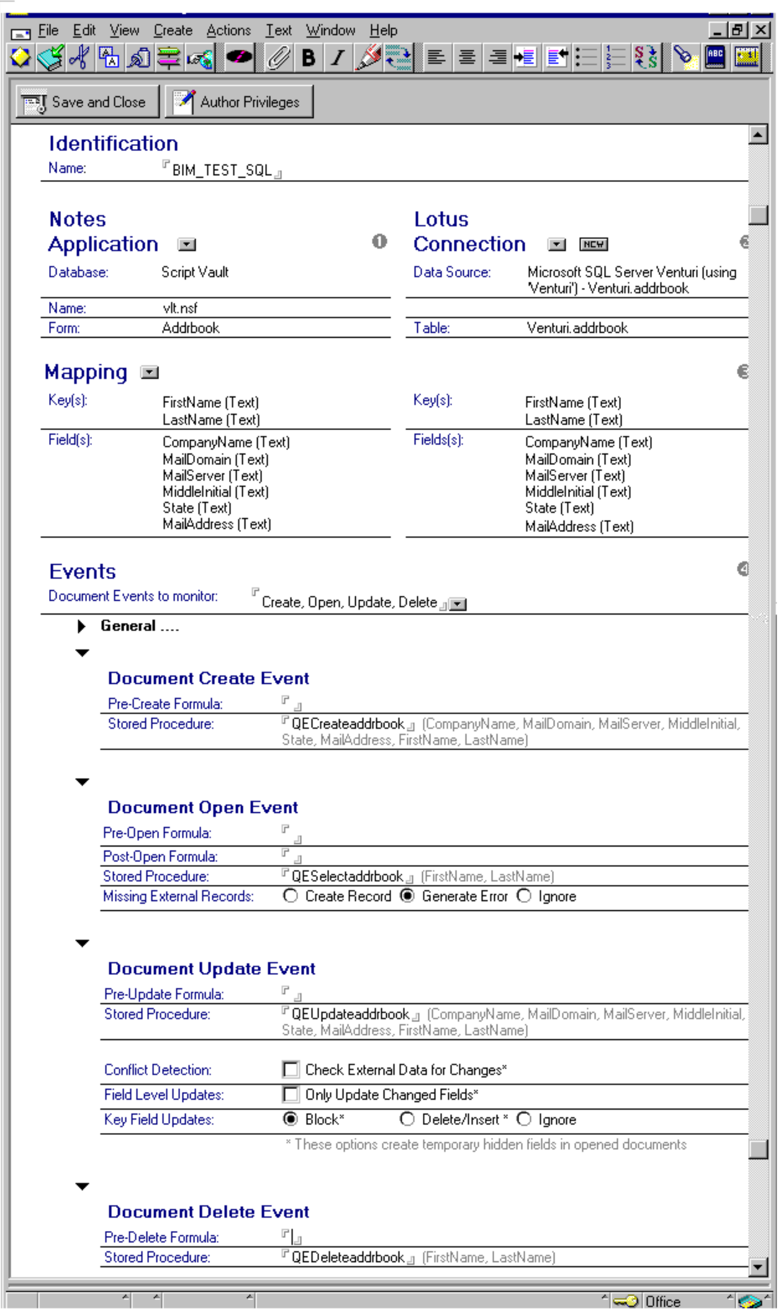
**Caution** When you use a stored procedure, field mapping order in the Activity must match the order of the stored procedure's parameters.

### Stored Procedure Example

Below is a set of stored procedures. These are followed by an example of how they should be entered in an LEI RealTime Activity document.

```
CREATE procedure QECreateaddrbook(

@CompanyName varchar(20),

@MailDomain varchar(20),

@MailServer varchar(20),

@MailAddress varchar(20),

@MiddleInitial varchar(1),
@State varchar(20),

@FirstName varchar(20),

@LastName varchar(20))

as

INSERT INTO Venturi.addrbook
(FirstName,MiddleInitial,LastName, MailDomain, MailServer,
MailAddress, CompanyName,State)

VALUES

(@FirstName, @MiddleInitial, @LastName, @MailDomain,
@MailServer, @MailAddress, @CompanyName, @State)

create procedure QEDeleteaddrbook (@FirstName varchar (20),
@LastName varchar (20)) as DELETE FROM addrbook WHERE
FirstName=@FirstName and LastName = @LastName

CREATE procedure dbo.QESelectaddrbook(@FirstName
varchar(20),@LastName varchar(20)) as

select FirstName, MiddleInitial, LastName, MailDomain,
MailServer, MailAddress, CompanyName, State from
Venturi.addrbook

WHERE  FirstName = @FirstName and LastName = @LastName

CREATE procedure dbo.QEUpdateaddrbook(

@CompanyName varchar(20),

@MailDomain varchar(20),

@MailServer varchar(20),

@MiddleInitial varchar(1),

@State varchar(20),

@MailAddress varchar(20),

@FirstName varchar(20),

@LastName varchar(20))

as

UPDATE Venturi.addrbook SET

MailDomain=@MailDomain, MailServer=@MailServer,
MailAddress=@MailAddress, CompanyName=@CompanyName,
State=@State

WHERE

FirstName=@FirstName and LastName=@LastName
```

The following illustration shows how these stored procedures would be referenced in a RealTime Activity document.



## Identification

| | |
|---|---|
| Name: | BIM_TEST_SQL |

**Notes Application**

| | |
|---|---|
| Database: | Script Vault |
| Name: | vlt.nsf |
| Form: | Addrbook |

**Lotus Connection**

| | |
|---|---|
| Data Source: | Microsoft SQL Server Venturi (using 'Venturi') - Venturi.addrbook |
| Table: | Venturi.addrbook |

**Mapping**

| | | | |
|---|---|---|---|
| Key(s): | FirstName (Text)<br>LastName (Text) | Key(s): | FirstName (Text)<br>LastName (Text) |
| Field(s): | CompanyName (Text)<br>MailDomain (Text)<br>MailServer (Text)<br>MiddleInitial (Text)<br>State (Text)<br>MailAddress (Text) | Fields(s): | CompanyName (Text)<br>MailDomain (Text)<br>MailServer (Text)<br>MiddleInitial (Text)<br>State (Text)<br>MailAddress (Text) |

**Events**

Document Events to monitor: Create, Open, Update, Delete

General ....

**Document Create Event**

| | |
|---|---|
| Pre-Create Formula: | |
| Stored Procedure: | QECreateaddrbook (CompanyName, MailDomain, MailServer, MiddleInitial, State, MailAddress, FirstName, LastName) |

**Document Open Event**

| | |
|---|---|
| Pre-Open Formula: | |
| Post-Open Formula: | |
| Stored Procedure: | QESelectaddrbook (FirstName, LastName) |
| Missing External Records: | Create Record [✓ Generate Error] Ignore |

**Document Update Event**

| | |
|---|---|
| Pre-Update Formula: | |
| Stored Procedure: | QEUpdateaddrbook (CompanyName, MailDomain, MailServer, MiddleInitial, State, MailAddress, FirstName, LastName) |
| Conflict Detection: | ☐ Check External Data for Changes* |
| Field Level Updates: | ☐ Only Update Changed Fields* |
| Key Field Updates: | [✓ Block*] Delete/Insert * Ignore |

* These options create temporary hidden fields in opened documents

**Document Delete Event**

| | |
|---|---|
| Pre-Delete Formula: | |
| Stored Procedure: | QEDeleteaddrbook (FirstName, LastName) |

## Microsoft SQL Server Data Types

The following tables show the correspondences of Microsoft SQL Server to LEI data types for various types of Activities.

- (p) – Indicates that if Allow Precision Loss is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not, an error is generated.

Refer to Chapter 5, "Introduction to Connectors" for more information about these tables.

### Execute

| *Microsoft SQL Server* | *Lotus Connector API (used by LEI)* |
|---|---|
| BIT | Int |
| TINYINT | Int |
| SMALLINT | Int |
| INT | Int |
| REAL | Float |
| FLOAT | Float |
| DECIMAL | Numeric |
| NUMERIC | Numeric |
| UNIQUEIDENTIFIER | Text (fixed length, bound = 39) |
| SMALLMONEY | Currency |
| MONEY | Currency |
| SMALLDATETIME | Datetime |
| DATETIME | Datetime |
| CHAR | Text (fixed length, bound <= 8000) |
| VARCHAR | Text (variable length, bound <= 8000) |
| TEXT | Text (variable length, unbounded) |
| NCHAR | Text (fixed length, bound <= 8000) |
| NVARCHAR | Text (variable length, bound <= 8000) |
| NTEXT | Text (variable length, unbounded) |
| BINARY | Binary (fixed length, bound <= 8000) |

*continued*

| *Microsoft SQL Server* | *Lotus Connector API (used by LEI)* |
|---|---|
| VARBINARY | Binary (variable length, bound <= 8000) |
| IMAGE | Binary (variable length, unbounded) |
| TIMESTAMP | Binary |

## Fetch

| *Microsoft SQL Server* | *Lotus Connector API (used by LEI)* | |
|---|---|---|
| BIT, TINYINT, SMALLINT, INT | Int | |
| TINYINT, SMALLINT, INT, REAL, FLOAT, DECIMAL (p, o), NUMERIC (p, o), SMALLMONEY, MONEY (p) | Float | |
| TINYINT, SMALLINT, INT, REAL (o), FLOAT (o), DECIMAL (p, o), NUMERIC (p, o), SMALLMONEY, MONEY | Currency | |
| TINYINT(p, o), SMALLINT (p, o), INT (p, o), REAL (p, o), FLOAT (p, o), DECIMAL (p, o), NUMERIC (p, o), SMALLMONEY(p, o), MONEY (p, o) | Numeric | |
| SMALLDATETIME, DATETIME (p) | Datetime | |
| CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT, BINARY, VARBINARY, IMAGE, TIMESTAMP UNIQUEIDENTIFIER | Text (o) | |
| CHAR, VARCHAR, TEXT, BINARY, VARBINARY, IMAGE, TIMESTAMP | Binary (o) | BLOB |
| SMALLDATETIME, DATETIME (p) | | Datetime list |
| TINYINT, SMALLINT, INT, REAL, FLOAT, DECIMAL (p, o), NUMERIC (p, o), SMALLMONEY, MONEY (p) | | Number list |
| CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT, BINARY, VARBINARY, IMAGE, TIMESTAMP UNIQUEIDENTIFIER | | Text list |

## Insert/Update

| *Lotus Connector API (used by LEI)* | | *Microsoft SQL Server* |
|---|---|---|
| Int | | BIT, TINYINT (p, o), SMALLINT (p, o), INT, REAL (p), FLOAT, DECIMAL (p), NUMERIC (p), SMALLMONEY, MONEY, CHAR (o), VARCHAR (o), TEXT, NCHAR (o), NVARCHAR (o), NTEXT |
| Float | | TINYINT (p, o), SMALLINT (p, o), INT (p, o), REAL (p, o), FLOAT, DECIMAL (p, o), NUMERIC (p, o), SMALLMONEY (p, o), MONEY (p, o), CHAR (o), VARCHAR (o), TEXT, NCHAR (o), NVARCHAR (o), NTEXT |
| Currency | | INT (p, o), REAL (p, o), FLOAT (p, o), DECIMAL (p), NUMERIC (p), SMALLMONEY (p, o), MONEY, CHAR (o), VARCHAR (o), TEXT, NCHAR (o), NVARCHAR (o), NTEXT |
| Numeric | | INT (p, o), REAL (p, o), FLOAT (p, o), DECIMAL (p, o), NUMERIC (p, o), CHAR (o), VARCHAR (o), NCHAR (o), NVARCHAR (o) |
| Datetime | | SMALLDATETIME (p), DATETIME |
| Text | | CHAR (o), VARCHAR (o), TEXT, NCHAR (o), NVARCHAR (o), NTEXT, BINARY (o), VARBINARY (o), IMAGE |
| Binary: | BLOB | CHAR (o), VARCHAR (o), TEXT, BINARY (o), VARBINARY (o), IMAGE |
| | number list | TINYINT (p, o), SMALLINT (p, o), INT (p, o), REAL (p, o), FLOAT, DECIMAL (p, o), NUMERIC (p, o), SMALLMONEY (p, o), MONEY (p, o), CHAR (o), VARCHAR (o), TEXT, NCHAR (o), NVARCHAR (o), NTEXT |
| | datetime list | SMALLDATETIME (p), DATETIME |

## Create

| *Lotus Connector API (used by LEI)* | | *Microsoft SQL Server* |
|---|---|---|
| Int | | INT, TINYINT, SMALLINT, or BIT |
| Float | | FLOAT, or REAL |
| Currency | | MONEY, SMALLMONEY |
| Numeric | | NUMERIC (prec, scale) |
| Datetime | | DATETIME or DATETIME4 |
| Text | | CHAR (fixed, len<=8000), VARCHAR (variable, len<=8000), or TEXT (len>8000) |
| Binary: | BLOB | BINARY(fixed, len<=8000), VARBINARY (variable, len<=8000), or IMAGE (len>8000) |
| | composite | CHAR (fixed, len<=8000), VARCHAR (variable, len<=8000), or TEXT (len>8000) |
| | number list | FLOAT |
| | datetime list | DATETIME |
| | text list | CHAR (fixed, len<=8000), VARCHAR (variable, len<=8000), or TEXT (len>8000) |

## OLE DB Connector Properties

See Appendix B of the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of properties for the OLE DB Connector.

# Chapter 14
# Sybase Connector

This chapter provides information about the Sybase Connector.

## Introduction to the Sybase Connector

Use a Sybase Connector to connect LEI to a Sybase database.

Refer to the *LEI Domino Connectivity and Installation Guide* for information about connecting to a Sybase database.

## Terminology

The table below lists LEI terms and the corresponding terms in Sybase.

| *LEI* | *Sybase* |
|---|---|
| Server | Server |
| Database | Database |
| Metadata | Table |
| Index | Index |
| Field | Column |
| Record | Row |

## Supported Characteristics

The following list captures supported characteristics of the Sybase Connector.

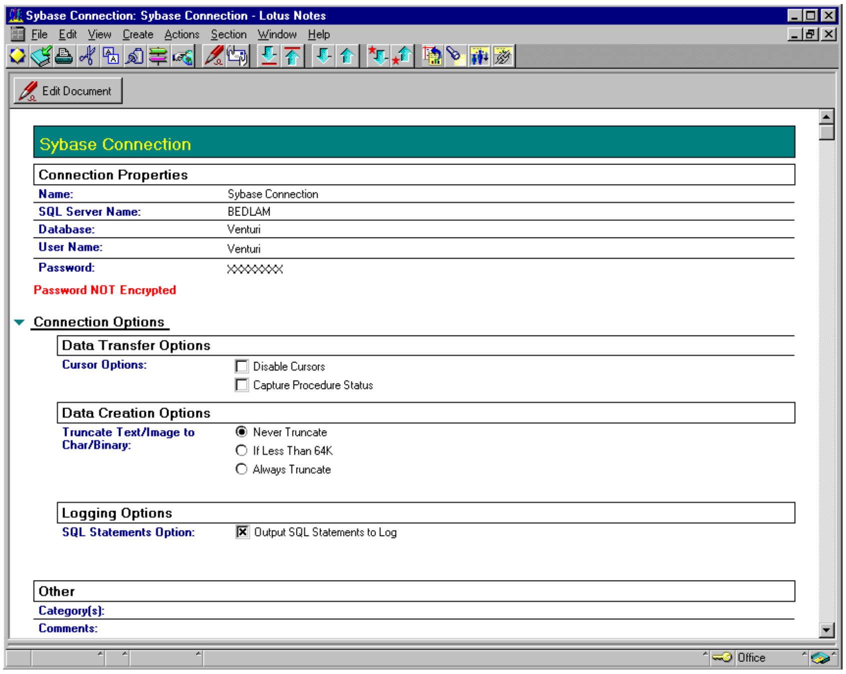
For a definition of the terms used in this section, see “Supported Connector Characteristics – List of Terms” in Chapter 5 of this manual.

- Writeback support – Full
- Writeback index – Index required. Use the command “create unique index <indexname> on <tablename> (<column1>, <column2>, …, <columnN>)”
- Statement syntax – Full Transact-SQL syntax, including stored procedures
- Condition syntax – Valid when placed in the following Sybase statement: SELECT … WHERE <condition> ORDER BY ..
- Array transfer – None
- Actions supported – Clear, Reset, Truncate
- Catalog types – Database (no connection required), Metadata, Index, Field, Procedure, Parameter
- Create types – Metadata, Index
- Drop types – Metadata, Index

**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for a list of supported properties for each Connector. Also see sections “Common Connector Properties” and “LCConnection Properties” in the LCConnection Class chapter of that same manual.

## Sybase Connection Document

The Sybase Connection Document, shown below, defines a Connection to a Sybase database and the Server on which it can be found. Choose Create – Connection – Sybase in the LEI Administrator, or click Create Connection from the Navigator and select Sybase to access the Sybase Connection Document.


Sybase Connection: Sybase Connection - Lotus Notes
File Edit View Create Actions Section Window Help
Edit Document
Office

Sybase Connection

Connection Properties

| | |
|---|---|
| Name: | Sybase Connection |
| SQL Server Name: | BEDLAM |
| Database: | Venturi |
| User Name: | Venturi |
| Password: | XXXXXXXX |

Password NOT Encrypted

Connection Options

Data Transfer Options

Cursor Options: ☐ Disable Cursors ☐ Capture Procedure Status

Data Creation Options

Truncate Text/Image to Char/Binary: ● Never Truncate ○ If Less Than 64K ○ Always Truncate

Logging Options

SQL Statements Option: ☒ Output SQL Statements to Log

Other

Category(s):

Comments:

## Connection Properties

Sybase Connection properties are described in the following table.

| *Field* | *Description* |
|---|---|
| Connector Name | Enter a unique name that identifies the Connector. |
| SQL Server Name | Enter the name of the Sybase SQL Server where the database is located. |
| Database | Enter the name of the Sybase SQL Server database. |
| User Name | Enter the user name required to log in to the Sybase SQL Server database. |
| Password | Enter the password associated with the user name above. |

## Connection Options

Sybase Connection Options let you specify data transfer, data creation, and logging options. The fields in the Sybase Connection Options section are described below.

### Data Transfer Options

| *Field* | *Description* |
|---|---|
| Disable Cursors | Disable the use of Sybase cursors. These cursors lock the table when producing a writeback result set (such as through replication). When modifying the table outside the cursor, such as through an insert or timestamp replication update, the operation can deadlock against the cursor. If this behavior occurs, simulate cursors through keyed operations by selecting this option. |
| Capture Procedure Status | A Sybase stored procedure can generate a result set or return a status code. When a non-zero code is returned, this may signal an error. The default behavior is to interpret the status as a 1-row, 1-column result set. Selecting this option interprets the status code separately, not as a result set, but as an error indicator (non-zero is error, and zero is success). |

### Data Creation Options

The Sybase Connection Data Creation Options are described in the table below. The Truncate option is used when the database is a destination database and a table is being created.

| *Field* | *Description* |
|---|---|
| Truncate Text/Image to Char/Binary | The long Sybase types Text and Image are inefficient and should only be used when necessary. When a Sybase table is created, Text and Binary columns longer than 254 characters are created as Sybase Text or Image columns. Use this option to demote Text and Image types to Char and Binary types. |
| Never Truncate | This option prevents truncation of Text and Image types. |
| Less than 64K | This option causes the truncation of all LEI Text and Binary types that are less than 64 kilobytes in length, but does not truncate lengths of 64 kilobytes or greater. |
| Always Truncate | This option causes the truncation of LEI Text and Binary types to Sybase Char and Binary types. |

### Logging Options

The Logging Options provide a feature for including SQL statements in the Activity log.

| *Field* | *Description* |
|---|---|
| Output SQL Statements to Log | Select this option to include all SQL statements generated during the execution of the Activity in the Activity's log. This can be helpful when you need to troubleshoot the Activity. |

## Sybase Operational Considerations

This section describes several important considerations for using the Sybase Connector:

### Replication

A Sybase SQL Server table that is the destination for a Replication Activity must have a unique index. Submit the following command through ISQL or another Sybase front end to create the index before attempting the Replication:

```
create unique index <indexname> on <tablename> (<col1>,
<col2>,…, <colN>)
```

Note that you can use just one column or many columns, but for efficient replication, the replication keys should be included in the index.

To remove an index, use the command:

```
drop index <tablename>.<indexname>
```

When doing Timestamp Replication using the LEI Replication Activity, a Sybase timestamp type column (an internal type) cannot be used as the timestamp. It must be a datetime or smalldatetime type.

### Data Insert

In Activities that insert data into a Sybase table, performance improves if the target table does not contain columns of type Text or Image. If the target is automatically created by LEI to contain these types, and the columns do not have to be Text or Image, set the Connection option Truncate Text/Image to Char/Binary to Always Truncate.

## Calling Sybase Stored Procedures

Wherever procedures are allowed in LEI, a Sybase stored procedure may be used. This includes RealTime Notes Activities, as the destination of a Direct Transfer, or within an LSX script. The stored procedure being called must have its parameters set up correctly for the call.

Input values are provided to Sybase procedures as named parameters. This requires that the parameters in Sybase use the same names as the LEI fields being provided as input values, The inputs being provided include key values when being used in the context of a keyed operation (selection, update, or delete context), and data values when relevant (insert or delete context). The input value datatypes are selected as the closest match to the datatype in the LEI system, and will be converted by Sybase to the procedure parameter defined datatypes, as long as the conversion is supported by Sybase.

Any output from a Sybase stored procedure must be returned through a final SELECT statement in the procedure. The result set of this selection will become the result set produced by the stored procedure.

The following is an example of a Sybase stored procedure. This is the format that would be required for the Open event of a RealTime Notes Activity assuming that the key field is called NUMBER1 and the data fields are called NUMBER2, TEXT1, and TEXT2. In this context, the key field is the input parameter, and the result set is expected to include the data fields followed by the key field.

```
create procedure OpenProc @NUMBER1 int

as select NUMBER2, TEXT1, TEXT2, NUMBER1 from tablename

where NUMBER1 = @NUMBER1
```

## Sybase Data Types

The following tables show the correspondences of Sybase to LEI data types for various types of Activities.

- (p) – Indicates that if Allow Precision Loss is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not, an error is generated.

For more information on the organization of these tables, see the section "Notes on Connector Data Type Conversion Tables" in Chapter 5.

### Execute

| *Sybase* | | *LEI* |
|---|---|---|
| BIT | | Int |
| TINYINT | | Int |
| SMALLINT | | Int |
| INT | | Int |
| REAL | | Float |
| FLOAT | | Float |
| DECIMAL | prec-scale<=9, scale<=0 | Int |
| DECIMAL | prec <=15 | Float |
| DECIMAL | other | Numeric |
| NUMERIC | prec-scale<=9, scale<=0 | Int |
| NUMERIC | prec <=15 | Float |
| NUMERIC | other | Numeric |
| MONEY | | Currency |
| SMALLMONEY | | Currency |
| DATETIME | | Datetime |
| SMALLDATETIME | | Datetime |
| CHAR | | Text (fixed length, bound <= 256) |
| NCHAR | | Text (fixed length, bound <= 256) |
| VARCHAR | | Text (variable length, bound <= 256) |

*continued*

| *Sybase* | *LEI* |
|---|---|
| NVARCHAR | Text (variable length, bound <= 256) |
| TEXT | Text (variable length, unbounded) |
| BINARY | Binary (fixed length, bound <= 256) |
| VARBINARY | Binary (variable length, bound <= 256) |
| IMAGE | Binary (variable length, unbounded) |

## Fetch

| *LEI* | | *Sybase* |
|---|---|---|
| Int | | BIT, TINYINT, SMALLINT, INT, REAL (o), FLOAT (p, o), DECIMAL (p, o), NUMERIC (p, o), MONEY (p, o), SMALLMONEY (p,o) |
| Float | | BIT, TINYINT, SMALLINT, INT, REAL, FLOAT, DECIMAL (p), NUMERIC (p), MONEY (p), SMALLMONEY (p) |
| Currency | | BIT, TINYINT, SMALLINT, INT, REAL (o), FLOAT (o), DECIMAL (p, o), NUMERIC (p, o), MONEY, SMALLMONEY |
| Numeric | | BIT, TINYINT(p, o), SMALLINT (p, o), INT (p, o), REAL (p, o), FLOAT (p, o), DECIMAL (p, o), NUMERIC (p, o), MONEY (p, o), SMALLMONEY (p, o) |
| Datetime | | SMALLDATETIME, DATETIME |
| Text (o) | | CHAR, NCHAR, VARCHAR, NVARCHAR, TEXT, BOUNDARY, SENSITIVITY, BINARY, VARBINARY, IMAGE |
| Binary (o) | BLOB | CHAR, NCHAR, VARCHAR, NVARCHAR, TEXT, BOUNDARY, BINARY, VARBINARY, IMAGE |
| | non-BLOB | invalid |

## Insert/Update

| *LEI* | | *Sybase* |
|---|---|---|
| Int | | BIT (p), TINYINT (p, o), SMALLINT (p, o), INT, REAL (p), FLOAT, DECIMAL (p), NUMERIC (p), MONEY, SMALLMONEY |
| Float | | BIT (p), TINYINT (p, o), SMALLINT (p, o), INT (p, o), REAL (p, o), FLOAT, DECIMAL (p, o), NUMERIC (p, o), MONEY (p, o), SMALLMONEY (p, o) |
| Currency | | BIT (p), TINYINT (p, o), SMALLINT (p, o), INT (p, o), REAL (p), FLOAT (p, o), DECIMAL (p), NUMERIC (p), MONEY, SMALLMONEY |
| Numeric | | BIT (p), TINYINT (p, o), SMALLINT (p, o), INT (p, o), REAL (p), FLOAT (p, o), DECIMAL (p, o), NUMERIC (p, o), MONEY (p, o), SMALLMONEY (p, o) |
| Datetime | | SMALLDATETIME, DATETIME |
| Text | | CHAR (o), NCHAR (0), VARCHAR (o), NVARCHAR (o), TEXT, BOUNDARY (o), BINARY (o), VARBINARY (o), IMAGE |
| Binary | any | CHAR (o), NCHAR (0), VARCHAR (o), NVARCHAR (o), LONGCHAR, TEXT, BOUNDARY (o), SENSITIVITY (o), BINARY (o), VARBINARY (o), LONGBINARY, IMAGE |
| | number list | BIT (p), TINYINT (p, o), SMALLINT (p, o), INT (p, o), REAL (p, o), FLOAT, DECIMAL (p), NUMERIC (p), MONEY (p), SMALLMONEY (p) |
| | datetime list | SMALLDATETIME, DATETIME |

## Create

| *LEI* | | *Sybase* |
|---|---|---|
| Int | | INT, NUMERIC (prec), or BIT |
| Float | | FLOAT, NUMERIC (prec), or REAL |
| Currency | | MONEY, SMALLMONEY |
| Numeric | | NUMERIC (prec, scale) |
| Datetime | | SMALLDATETIME, DATETIME |
| Text | | CHAR (fixed, len<=256), NCHAR (fixed, len<=256), VARCHAR (variable, len<=256), NVARCHAR (variable, len<=256), TEXT (len>256) |
| Binary | BLOB | BINARY(fixed, len<=256), VARBINARY (variable, len<=256), or IMAGE (len>256) |
| | composite | CHAR (fixed, len<=256), NCHAR (fixed, len<=256), VARCHAR (variable, len<=256), NVARCHAR (variable, len<=256), TEXT (len>256) |
| | number list | FLOAT |
| | datetime list | SMALLDATETIME, DATETIME |
| | text list | CHAR (fixed, len<=256), NCHAR (fixed, len<=256), VARCHAR (variable, len<=256), NVARCHAR (variable, len<=256), TEXT (len>256) |

# Chapter 15
# Text Connector

This chapter provides information about the Text Connector.

## Introduction to the Text Connector

The Text Connector allows data transfer between text files and all LEI-supported Connectors. You define formats for the input (source) and output (destination) data in a file, called the ZID file. The ZID file is referenced by file name from the Text Connection form, or you can specify the actual ZID file contents entirely within the Text Connection form. You can also select various options for processing the source and destination data and can select internationalization of days of the week and months of the year.

## Supported Activities

The Text Connector does not support browsing. It does not support selection criteria, keyed operations, or conditions. As a result it cannot be used with Replication, Archive, or RealTime Activities. In addition, the command statement has no effect: all records are retrieved for relevant Activities (including Direct Transfer and Polling).

**Note** The Text Connector does not support metadata browsing.

## Terminology

The table below lists LEI terms and the corresponding terms used by the Text Connector.

| *LEI* | *Text* |
|---|---|
| Server | (Not applicable) |
| Database | File name |
| Metadata | Replacement substrings |
| Index | (Not applicable) |
| Field | Field, as defined in the information description (ZID) area |
| Record | Record, as defined by record delimiter or record length |

## Supported Characteristics

The following list captures supported characteristics of the Text Connector.

For a definition of the terms used in this section, see "Supported Connector Characteristics – List of Terms" in Chapter 5 of this manual.

- Writeback support – N/A
- Writeback index – N/A
- Statement syntax – N/A – use ZID file instead
- Condition syntax – N/A
- Array transfer – None
- Actions supported – Clear, Reset, Truncate
- Catalog types – None
- Create types – None
- Drop types – None

**Note** All Connections have a set of properties with values that can be assigned and retrieved. See Appendix C in the Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide for a list of supported properties for each Connector. Also see sections "Common Connector Properties" and "LCConnection Properties" in the LCConnection Class chapter of that same manual.

The Text Connector supports a limited Select operation with minimal properties and no keys. This allows it to be used in combination with the Order MetaConnector as the source of a one-way non-timestamp Replication Activity.

**Note** When using long file names, include the filename as is (with any embedded spaces). Do not put quotation marks around the file name, since the quotation marks will be considered part of the file name.

## Text Connection Document

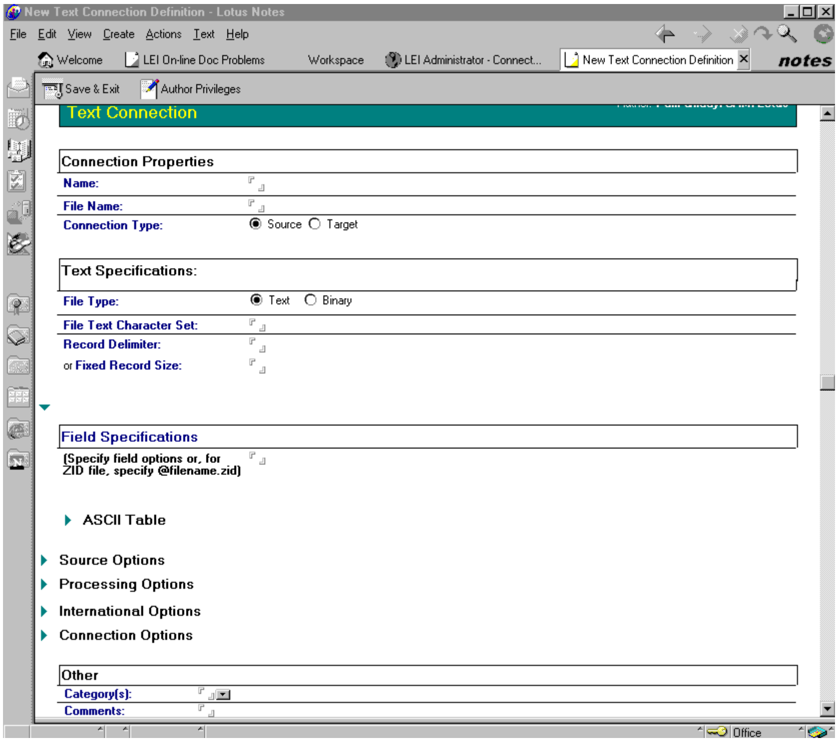
The Text Connection forms are used to define the location and format of the text data. Text Connections are defined as source or destination processing options, and then define the location and format of the data in its destination. The format and basic methodologies are the same for each specification.

The following sections describe the fields and areas on the Text Connection form. In summary, they are:

- Connection Properties (Definition)
- Text Specifications
- Field Specifications
- Source & Target Options
- Processing Options
- International Options
- Text Connection Options

## Connection Properties

Access the Text Connection form from the LEI Administrator. Click the Create Connection icon in the Navigator and select Text, or from the menu choose Create – Connection – Other, then select Connection/Text from the list.


New Text Connection Definition - Lotus Notes
File Edit View Create Actions Text Help
Welcome
LEI On-line Doc Problems
Workspace
LEI Administrator - Connect...
New Text Connection Definition
notes
Save & Exit
Author Privileges
Office

Text Connection

Connection Properties

Name:

File Name:

Connection Type: [✓ Source] Target

Text Specifications:

File Type: [✓ Text] Binary

File Text Character Set:

Record Delimiter:

or Fixed Record Size:

Field Specifications

(Specify field options or, for ZID file, specify @filename.zid)

ASCII Table

Source Options

Processing Options

International Options

Connection Options

Other

Category(s):

Comments:

The Connection Name and File Name are required information.

| *Field* | *Description* |
|---|---|
| Name | Enter a name for the Connection. |
| File Name | The path of the source or destination text, for example, C:\data\datafile.txt. When using long filenames, include the filename as is (with any embedded spaces). Do not use quotation marks in the filename since they will be considered part of the filename. |

## Text Specifications

Text files can be composed of either fixed-length or variable records. Fixed-length records are determined by a user-defined decimal value that specifies the actual record length. Variable length records are determined by a user-specified delimiter in character or hexadecimal notation, for example, a carriage return/line feed.

The Text Field Specification describes either the input or output data, depending on whether you are defining a source or destination form. For either form enter the following information:

**File Type —** Select either the Text or Binary radio button to indicate the type of file you are describing in this form.

By default, Text processes files in text mode. Text mode translates each carriage return/line-feed combination to mean a new record. In certain circumstances, when records are not delimited by carriage return, linefeed combinations, you might want the Text Connection to process the data exactly as is with no text mode translations, and would select binary as the file type.

**Record Delimiter —** If the data records are variable in length, enter the delimiter you are using. The default is \n, indicating a new line or carriage return.

### File Text Character Set

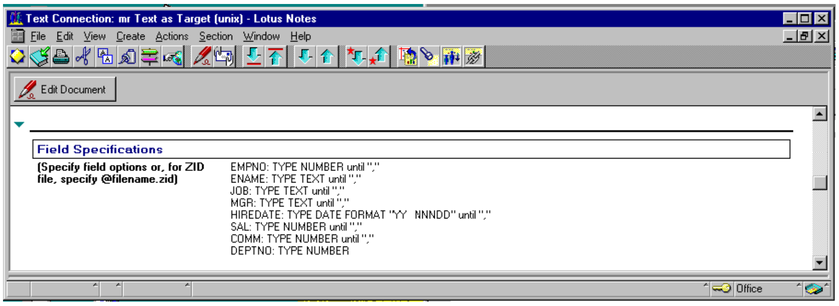
If you want to override the default character set of the underlying Connector, enter the overriding character set in this field. For example, enter "CP392" if you wish to override the default character set with the CP392 character set.

Character sets and default code pages are described in Appendix C.

The default character set is the character set that is native to the machine on which you are working.

## Field Specifications

Either reference the ZID file by name (using the syntax: @filepath\filename.zid) or specify the actual field options in this section of the Text Connection document.


Text Connection: mr Text as Target (unix) - Lotus Notes
File Edit View Create Actions Section Window Help
Edit Document
Office

Field Specifications

(Specify field options or, for ZID file, specify @filename.zid)

```
EMPNO: TYPE NUMBER until ","
ENAME: TYPE TEXT until ","
JOB: TYPE TEXT until ","
MGR: TYPE TEXT until ","
HIREDATE: TYPE DATE FORMAT "YY   NNNDD" until ","
SAL: TYPE NUMBER until ","
COMM: TYPE NUMBER until ","
DEPTNO: TYPE NUMBER
```

Notice that an ASCII table is provided to assist you in creating definition strings. The ASCII table lists the decimal, hexadecimal, and character equivalents for the ASCII character set. Supported escape sequence characters are indicated in red text.

## Field Options and Text Information Definition Syntax

Text fields can be read and written in fixed positions or in floating/delimited positions. Fixed position fields are determined by user-defined start and end or length column definitions. Floating/delimited fields are determined by placement before or after other fields in the ZID file definition.

For input text data, field definitions provide a logical name for data items, decipher special format issues, and specify ultimate destination attributes. For output text data, field definitions define number and date formatting, justification, and literal string values. Allowable text formats are:

- Text (converts to and from Notes simple text or rich text)
- Numbers (including currency symbols)
- Dates (including time components)

You create a field entry in the ZID file or in the Text Field Specifications area for each field that you want to process.

Field entries are in the form:

```
<fieldname> TYPE <datatype> <field option <argument>>
```

where:

**`<fieldname>`**

**For source Connection access:**

**`<fieldname>`** is the database field name assigned to the data within LEI. The **`<fieldname>`** is required.

Parentheses around any field name indicate a “placeholder” that you do *not* want assigned as a field within LEI in a destination database record. For example, if you are reading a file that includes a field called Publisher which you do not need, use the expression (Publisher) in the fieldman to omit the data.

**For target Connection access:**

**`<fieldname>`** is the database field name to be extracted to the destination text. If the field name does not exist in the specified input database, then a *null value* is written to the output file. The **`<fieldname>`** is required.

Use parentheses around any field name that you do *not* expect to be present in the input database. This allows the provisioning of literal strings to the output file.

**`TYPE`** defines the field data type as specified by **`<datatype>`**. This definition refers to the field that is being translated. You can specify any of the following values for **`<datatype>`**. The TYPE is required.

| *<datatype>* | *Description* |
|---|---|
| TEXT | Alphanumeric text. |
| NUMBER [.n] | Numeric text. The [.n] specifies a decimal place when a decimal is NOT present in the source data.<br>If the [.n] is specified, the incoming data should NOT include the decimal point. For example, input data 123.45 would fail whereas 12345 would be formatted in the output as 123.45. The “.n” can be any value in the range 0 to 15. The default is 0, or no decimal digits. |
| DATETIME | A date/time expression. The date value must be suitable for conversion in the native machine environment. |
| RICHTEXT | Alphanumeric text. |

Above, <field option> is the name of a descriptive field option which provides additional information about the field being read or written. The <field option> is not required.

Some field options also use <arguments> to further define the data field.

You can define multiple field options/arguments for each input field. Simply leave a blank space between any field option or argument.
For text input, the location of the field in the input data is critical.

The location of an input field is either fixed (at the same place in every input record) or floating (input data is a stream of fields separated by special delimiters).

**Note** You can add ZID comment entries to a ZID script by beginning a line (the first position in a line) in the ZID script with a semicolon. For example, the following statement is a valid ZID comment entry:

```
; This ZID file describes the monthly update
```

## Defining Source Files

This section provides examples showing how to define test and binary files that are organized in several different ways: single line, multiple line, fixed length fields, and floating field format. The examples are:

- Text input files – Single line per record (both fixed and floating fields formats)
- Text input files – Multiple lines per record
- Binary input files – Fixed length records

The ZID field definition has the following syntax:

```
<fieldname> TYPE <datatype> <field option <argument>>
```

### Text Input Files – Single Line per Record

Data in a text file can be organized in several ways. In text files, a record consists of one or more fields terminated by a unique character or set of characters. Usually a record is simply a single "line" of text terminated by a new-line-character. This one record per line format is the most common and is easily manipulated using a Text Connection. The following example shows input data organized in the one record per line format.

```
IDNO-----NAME--------------TEL-------BALANCE----PAYMENT----DAT
E

566238744Carol Ann Wilson
10255577460000400009800000030000010-15-93

012358743Nathan Varberg
10155523450000030000000000025000010-11-93

524135698David Fein
10155532220000355995500000102550009-30-93
```

This example of a customer record includes a unique ID number, the customer name, telephone number, balance due, payment, and date. We use this input data as the basis for the examples in the next sections.

## Fixed Field Version

Using the data in the above sample input file as an example, the following example builds the ZID field entries necessary to process the data using LEI. For simplicity, all fields in this example are fixed, that is, they are all in the same location in every record. In addition, for purposes of visual clarity in this example, initial caps indicate entries you make, specific to your data.

In the following example, the first line denotes the field named Type as Customer. The first line illustrates the use of the VALUE field option for inserting literal strings into fields. This parameter inserts "Customer" into the Type field.

The first data field (the second line) in our example is a customer ID number (labeled IDNO). The IDNO field could be described as a number or a text field. Because the value here is for identification purposes and does not represent a quantity or magnitude, we define the IDNO field as text. In this example, the IDNO field always begins in column 1 (the first position in the record) and always ends in column 9. The Connector inserts this data into the field named IDNO.

The ZID field description in our first example look like this:

```
Type: TYPE TEXT VALUE "Customer"

Idno: TYPE TEXT START 01 END 9

Name: TYPE TEXT START 10 END 29

Tel:  TYPE TEXT START 30 END 39

Balance: TYPE NUMBER.2 START 40 END 50

Payment: TYPE NUMBER.2 START 51 END 61

Date: TYPE DATE START 62 END 67 FORMAT MM.DD.YY
```

**Note** NUMBER.2 is the correct format as long as the input data has no decimal point in it. If there is a decimal in the input data, then do not use the [.2] formatting.

The START and END field options with their corresponding numeric values describe the beginning and ending position of the input fields (note that you could use the WIDTH or LENGTH field options instead of the END field option). The decimal place (.2) notation after the TYPE NUMBER specification tells the Text Connection how many decimal places are implied in the input numeric value. If you do not configure the decimal place notation, the Text Connection copies the input numeric value as an integer.

Creating ZID field entries for date and time values is more complex than simple TEXT or NUMBER types. Dates and times come in various formats with unusual field separators, prefixes, and suffixes. The Text Connection allows for these various formats, but requires that you describe the input date format with a special FORMAT field option specification. Notice the format specified in the DATE field in our example.

The FORMAT field option identifies the various components of an input date and/or time field. In this example, M designates a month field, D designates a day field, and Y designates a year field. Refer to the FORMAT field option for more information on time and data formats.

### Floating Field Version

Another typical format for input data is the floating field format. A field is considered to be floating if its position is determined by the end of the field immediately before it in the input record. The example below shows data from our fixed field example as a floating field format file. Each field is separated from the subsequent field by a comma:

```
IDNO,NAME,TEL,BALANCE,PAYMENT,DATE

566238744,Carol Ann Wilson,1025557746,4000098,30000,10-15-93

012358743,Nathan Varberg,1015552345,300000,25000,10-11-93

524135698,David Fein,1015553222,3559955,102550,09-30-93
```

For the floating field version, configure the ZID field description as follows:

```
Type: TYPE TEXT VALUE "Customer"

Idno: TYPE TEXT UNTIL ","

Name: TYPE TEXT UNTIL ","

Tel:  TYPE TEXT UNTIL ","

Balance: TYPE NUMBER.2 UNTIL ","

Payment: TYPE NUMBER.2 UNTIL ","

Date: TYPE DATE FORMAT MM.DD.yy
```

The IDNO field begins in column 1 and continues until a comma is encountered in the input data. Subsequent fields are processed similarly. In the DATE field specification, notice that the last field in the record does not have an UNTIL specification. The last field in a record does not need an UNTIL specification because scanning for the end of the last value in a record always stops at the end of the data record.

## Text Input Files – Multiple Lines per Record

The Text Connection can also read input records from multiple contiguous lines of text. A multiple line per record format requires that you specify an input record delimiter (a unique character or string of characters) to denote the end of one input record and the beginning of the next input record. Specify the input record delimiter in the Text Connection. The record delimiter string can be up to 256 characters long. Usually the input record delimiter is a single character such as a form feed (\f) character or a paragraph symbol (¶).

The following example shows how the first three records from the single line example look in multi-line format. The input delimiter is a single character paragraph symbol.

```
IDN:566238744

Nm:Carol Ann Wilson

Tel:1025557746

Bal:4000098

Pmt:30000

Date:10-15-93

¶

IDN:012358743

Nm:Nathan Varberg

Tel:1015552345

Bal:300000

Pmt:25000

Date:10-11-93

¶

IDN:524135698

Nm:David Fein

Tel:1015553222

Bal:3559955

Pmt:102550

Date:09-30-93
```

To translate this multi-lined data, configure the ZID field description as follows:

```
Type: TYPE TEXT VALUE "Customer"

Idno: TYPE TEXT START 5 UNTIL "\nNm:"

Name: TYPE TEXT UNTIL "\nTel:"

Tel:  TYPE TEXT UNTIL "\nBal:"

Balance: TYPE NUMBER.2 "\nPmt:"

Payment: TYPE NUMBER.2 "\nDate"

Date: TYPE DATE FORMAT MM.DD.yy
```

Notice the differences in this example compared to the single line per record example. Remember we are using a record delimiter of (0x14). We use the escape sequence (\0x14) to specify the special character record delimiter string in hexadecimal notation. (Refer to the ASCII/hexadecimal equivalence chart for these values.)

The UNTIL specifications are multiple character strings, also with a special escape sequence (\n) to allow description of the line feed characters that delimit the input field values. The sequence (\n) indicates a single new-line-character.

Remember, the input record now spans several input lines. The IDNO field begins in column 5, after the “IDN:” label, and continues until the Text Connection encounters the string “\nNm:”. The Text Connection processes subsequent fields similarly. Notice, as before, in the DATE field specification, the last field in the record does not have an UNTIL specification. The UNTIL specification is not needed because, when scanning for the end of the last value in a record, the Text Connector always stops at the end of the data record.

## Binary Input Files – Fixed Length Records

Data in binary files can only be organized as fixed length records. All records contain the same amount of data (number of bytes). With binary files, a record consists of one or more fields of a given, fixed, length. You specify the fixed size of input binary data records with the record delimiter specification in the Text Connection. Binary input files most closely resemble the text fixed field format. As with the text fixed field format, fields within binary files must exist at the same location and have the same length in every input record.

The next example shows a typical ZID field description for reading IBM host-originated data with Text Connection. Character data from IBM host computers is generally in EBCDIC format. This example shows how the Text Connection can translate EBCDIC characters to ASCII text characters during processing. Note that the Text Connection can also translate packed decimal data, another common format found in IBM host data, to number types.

```
Type: TYPE TEXT VALUE "Customer"

Idno: TYPE TEXT START 01 END 9 EBCDIC

Name: TYPE TEXT START 10 END 29 EBCDIC

Tel:  TYPE TEXT START 30 END 39 EBCDIC

Balance: TYPE PACKED.2 START 40 END 44

Payment: TYPE PACKED.2 START 45 END 59

Date: TYPE DATE START 60 END 67 FORMAT MM.DD.YY EBCDIC
```

## Defining Target Files

Output records can be single or multiple line text. You can generate multiple lines by designating carriage returns or line feeds as UNTIL strings.

Within the Text Connector, you can define a character string to indicate the end of each output record. The Text Connector then uses this text string to delimit records For example, whenever the Text Connector encounters the specific text string, it knows to end the current output record and begin another. The default record delimiter is a single new-line character (\n).

The following example depicts a sample ZID field definition for output:

```
(Idn): TYPE TEXT VALUE "IDN:"

Idno: TYPE TEXT START 5 UNTIL "\nNm:"

Name: TYPE TEXT UNTIL "\nTel:"

Tel:  TYPE TEXT UNTIL "\nBal:"

Balance: TYPE NUMBER.2 UNTIL "\nPmt:"

Payment: TYPE NUMBER.2 UNTIL "\nDate"

Date: TYPE DATE FORMAT MM.DD.yy
```

As shown in the first four lines of the example, when you output document data you use the record delimiter to define a character to mark the end of each record. In this example, the escape sequence is a paragraph symbol indicated by the hexadecimal notation \0x14.

The UNTIL specifications are multiple character strings with a special escape sequence (\n). These allow the definition of the line feed characters that delimit the input field values and, ultimately, create our multiple line output file.

Notice, as in the input example, in the DATE field specification, the last field in the record does not have an UNTIL specification. The Text Connector does not require the UNTIL specification because the record delimiter is specified and the Text Connector places the delimiter string at the end of each output record. Also note the decimal points indicated by the NUMBER.2 options used in the ZID field description.

The ZID field description produces an output file as follows:

```
IDN:566238744

Nm:Carol Ann Wilson

Tel:1025557746

Bal:40000.98

Pmt:300.00

Date:10-15-93

¶

IDN:012358743

Nm:Nathan Varberg

Tel:1015552345

Bal:3000.00

Pmt:250.00

Date:10-11-93

¶

IDN:524135698

Nm:David Fein

Tel:1015553222

Bal:35599.55

Pmt:1025.50

Date:09-30-93
```

## Using Field Options

The Text Connection allows you to use field options to describe the location of fields in both fixed and variable records and also to format the data. As described previously, include the desired field option on the far right side of the field entry.

Use the following field options to designate location of data in a file.

- START [nnn] or
  START [@fldname]     Fixed position field begin column
- WIDTH [nnn] or Fixed position field end column
- LENGTH
- END [nnn]     Fixed position field end column
- UNTIL [until_string_spec]     Floating field string delimiter

Use the following field options to format the text data:

- EBCDIC  EBCDIC to ASCII conversion
- FORMAT     Set date format or special field forma
- RIGHT  Output right justify
- SEPARATOR  Multiple-value list delimiter
- TRIM  Remove redundant white space
- VALUE  Default values for input fields

More detailed information on these field options follows.

### Specifying Field Location

START, WIDTH, END, and LENGTH numbers designate the starting and ending columns of a fixed size column of data in a text file. Use START and END when columns are aligned at a fixed column position rather than being separated by specific characters. For example, if the customer ID number (IDNO) is a text field that always begins in column 1 and ends in column 9, specify the following:

```
IDNO: TYPE TEXT START 01 END 09.
```

Values can also “float” within a specified column of white space. For example, the field option location description of START 1 WIDTH 9 indicates a value somewhere in the first nine columns of a record. This means that the data in these three records has, in each case, a value of “5”.

```
Column position

123456789

Record 1:  5

Record 2:  5

Record 3:  5
```

The START @fldname syntax allows you to reset the current (start) position within the input record to redefine an input area for use by other database fields. It allows you to access portions of a field or group of fields. For example:

```
WholeField:      TYPE TEXT UNTIL ","

FirstChar:       TYPE TEXT START @wholeField WIDTH 1
```

The WholeField is defined as a text field containing all data up to the first comma within the input record. The FirstChar field specification resets the current field position back to the first WholeField position and, since the width is 1, sets the FirstChar field with only the first position of the Whole-Field value. The LENGTH option can be used in place of WIDTH.

UNTIL describes the character string that is used to delimit or end an input or output text item. A field within a text file is terminated at the first occurrence of the UNTIL string. In the last example, the delimiter is a comma (,), and WholeField continues to the first occurrence of a comma in the record. The specified string can contain escape sequences, such as the following:

| | |
|---|---|
| \a | Alarm or bell character |
| \b | Backspace character |
| \f | Form feed character |
| \n | New line character |
| \t | Tab character |
| \r | Carriage return |
| \\ | Single backslash |
| \v | Vertical tab character |

\0bbb Octal conversion (where bbb is the octal number to convert to a character)

\0xhh Hexadecimal conversion (where hh is the hexadecimal number to convert to a character)

\nnn  Integer conversion (where nnn is the decimal number to convert to a character)

UNTIL is useful for fields that are separated by a single word or string. The Text Connector scans input or output text until it encounters the UNTIL string specification. The Text Connector scans until the end of the current logical record. However, if it does not find the specified UNTIL string, all data up to the very end of the logical record is placed in the corresponding named field.

UNTIL is useful for describing quoted values separated by commas. For example, the following statement can be delimited by specifying UNTIL "","":

```
"FIELD 1", "FIELD 2"
```

Sometimes you want to watch for one of several different individual characters that can indicate the end of a field. The Text Connector allows you to watch for a list of these individual characters with the UNTILLIST field option. The Text Connector determines the end of a field at the first occurrence of any individual character in the UNTILLIST string.

UNTILLIST is useful for describing comma-separated values or other data fields separated by a unique character. UNTILLIST can be coded with START, END, and WIDTH parameters, in which case the UNTILLIST string is only searched for within the indicated START, END, or WIDTH columns.

**Note**  If you need to watch for several characters together, rather than individually, use the UNTIL field option.

## Specifying Formatting Options

The Text Connector supports the following script options and the functions they represent. Use them by placing the option name at the right side of the field entry.

```
EBCDIC
```

### EBCDIC to ASCII Conversion

The EBCDIC field option converts input fields from EBCDIC to ASCII before adding the fields to a Notes document.

**Note**  When you use EBCDIC, you must use fixed column positions with your fields. Also, if your input is in EBCDIC format, you cannot use the UNTIL format notation.

**Example:** In this example, Text Connector reads EBCDIC data from a tape dump and converts it to LEI text.

```
;EBCDIC.ZID

Type: TYPE TEXT VALUE "Stock"

Part: TYPE TEXT START 1 WIDTH 12 EBCDIC

Quantity: TYPE TEXT START 13 WIDTH 3 EBCDIC

Description: TYPE TEXT START 16 WIDTH 24 EBCDIC
```

## FORMAT

Use this to set date format or special field format.

FORMAT specifies the format for an input or output field. The field can be one of several types: DATE, TEXT, or NUMBER. This section describes the use of the FORMAT field option with each type.

### FORMAT with TYPE DATE

Use the format_spec for DATE fields to identify the various components of a date and/or time field where, for dates:

| | |
|---|---|
| M | designates a month field |
| N | designates a month name (for example, January, Feb, April). A month name must be at least three characters long to guarantee uniqueness. |
| D | designates a day field |
| Y | designates a year field. A lower case 'y' indicates a two-digit 20th century year. The Text Connector adds 1900 to the year value. |
| J | indicates a Julian date |
| W | designates a weekday name (for example, Monday). This is used in output only and for times: |
| h | designates an hour field |
| m | designates a minutes field |
| s | designates a seconds field. |

A or P indicates use of a meridian indicator (AM or PM). For example, if a date field had the following form:

```
02xxxMAR0315
```

the ZID entry could be (unrecognized characters are disregarded, so that “x” is disregarded):

```
Important Date: TYPE DATE BEGIN 1 WIDTH 12 FORMAT
"YYxxxxxxMMDD"
```

This would transform the ImportantDate field to a date with the format, “03/15/02.”

**Note** Only Y or y, M, N, D, J, h, m, s are recognized for formatting. W designates a weekday (destination only). Other characters are ignored.

**Note** If you you specify a date type without a format, the date value will be converted as described in the “Year 2000” section of Appendix A.

### FORMAT with TYPE TEXT or NUMBER

Use the format_spec to conditionally change values from the actual input values. The syntax for conditional FORMAT specifications is:

```
FORMAT "a,b,c=X;d,e=Y;Z"
```

You can read the format specification above as:

IF the input field is equal to a, b, or c, then set value to X

ELSE if the input field is equal to d or e, then set value to Y

ELSE set value to Z

If a final ELSE clause (without compare values) is not given, then the input value is unchanged.

**Note** The comparison is for the length of the compare value only, not the length of the input field. Fields are “normalized” before comparison, so leading and trailing white space is not compared.

**Example:**

This example uses a variety of date and text formats for an employee record.

```
; Mon_Name2 ZID SCRIPT

;

Type: TYPE TEXT VALUE "Employees"

Name: TYPE TEXT UNTIL ","

HireDate: TYPE DATE FORMAT NNN-DD-YY UNTIL ","

Title: TYPE TEXT FORMAT "Pres,Pre=CEO,Man,Mgr=MGR,EMP" UNTIL
","

Date_Of_Birth: TYPE DATE FORMAT yy-MMDD  UNTIL \r\n

*
```

```
Claude Hopper,Mar-07-95,Pres,14-0904
Evander Holyfield,Jun-30-96,Man,69-0315
```

### RIGHT

Use this to right-justify output.

With output TEXT data, you can control whether the data is left or right justified. The default justification is left. To override this and make it right justification, you must do two things:

- Specify the LENGTH of the data on the output field.
- Specify RIGHT on the output field.

**Note** The default justification for number fields is right justification.

**Example:**

In this example, the Text Connector sets the length of CompanyName, the field to be output, to 40 and specifies right justification. This results in leading white space for company names of less then 40 characters in length.

```
; RIGHT ZID SCRIPT
;Type: TYPE TEXT VALUE "1. Customer Profile"
CompanyName:     TYPE TEXT  UNTIL \r\n LENGTH 40 RIGHT
CompanyAddress:  TYPE TEXT  UNTIL \r\n
CompanyCity:     TYPE TEXT  UNTIL \r\n
CompanyState:    TYPE TEXT  UNTIL \r\n
CompanyZip:      TYPE TEXT  UNTIL \r\n
CompanyPhone:    TYPE TEXT  UNTIL \r\n
CompanyFax:      TYPE TEXT  UNTIL \r\n
Body:            TYPE RICH UNTIL \r\n  SEP \r\n
```

### SEPARATOR

Use this to enable a multiple-value list delimiter.

For non-rich text fields, SEPARATOR or “SEP” describes the characters separating multiple-value list items. SEP is useful for describing Lotus Notes multiple-value “list-type” fields whose elements are separated by a unique character. Other database types might or might not support multiple-value list types.

**Note** For non-rich text fields, the SEPARATOR option specifies a set of characters that can separate multiple field elements.

For rich text fields, Text Connector creates new rich text lines at each occurrence of the SEPARATOR string. For example: If your separator string is “$$“, you would enter “$$“. The default line separator is carriage-return and line-feed characters, or “\r\n”. When the default is used, any combination of carriage-return and line-feed characters creates a new line.

**Note** For rich text fields, SEPARATOR specifies a string of characters that, in its entirety, separates rich text lines.

Escape sequences are allowed as follows:

| | |
|---|---|
| `\a` | `alarm or bell character` |
| `\b` | `backspace character` |
| `\f` | `form feed character` |
| `\n` | `new line character` |
| `\r` | `carriage return character` |
| `\t` | `tab character` |
| `\v` | `vertical tab character` |
| `\0bbb` | `octal conversion (where bbb is the octal number to convert)` |
| `\0xhh` | `hexadecimal conversion (where hh is the hexadecimal number to convert)` |
| `\nnn` | `integer conversion (where nnn is the decimal number to convert)` |

**Example:**

In this example, Text Connector defines the SEP character string as “$$,“ overriding the defaults of carriage-return and line-feed.

```
; NSEP ZID SCRIPT
Type:      TYPE TEXT VALUE "Letter"
Subject:   TYPE TEXT UNTIL "," VIEWKEY 1
Author:    TYPE TEXT UNTIL ","
Body: TYPE TEXT SEP "$$" UNTIL \0x0c\r\n
*
Painters Invade Monhegan Island, SWD,
LINE1$$LINE2$$LINE3$$LINE4
```

### TRIM

Use this to eliminate extra white space in your input data. When the TRIM field option is used, Text Connector removes leading, trailing, and multiple interspersed white space characters (blanks and tabs).

Example:

In this example, input data embedded in the ZID script has extra white space in it. Using the TRIM field option, Text Connector parses out the white space.

```
; TRIM ZID

Type:           type text value "Stock"

Part:      TYPE TEXT UNTIL "," viewkey 1 TRIM

Quantity:  TYPE TEXT UNTIL "\n"

*

Rubber      Hoses       ,125

Widgets and           Gidgets    ,255
```

The output will look like this:

```
Rubber Hoses, 125

Nuts and Bolts, 255
```

### VALUE

Use this to set default values for input fields.

The value_spec is the default value for an input field. This is useful for adding new fields to a Notes document that have no corresponding input field. For example, given the ZID descriptor:

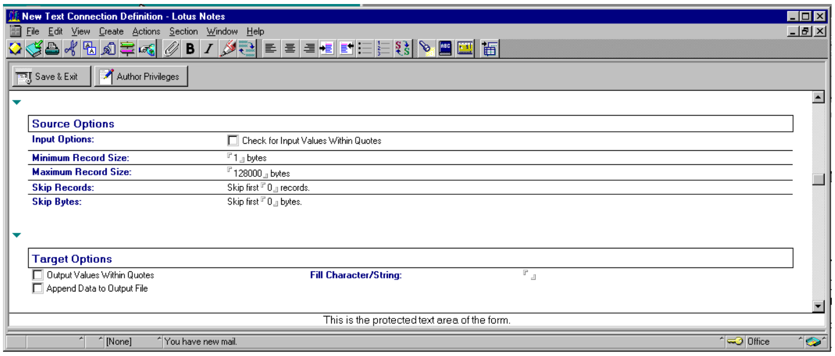
```
OFFICE_TITLE: TYPE TEXT VALUE "Originator"
```

Every document that the Text Connector adds to or updates from input would be given a new field, named OFFICE_TITLE, with a value of “Originator.”

## Source and Target Options

A Text Connection offers several options that specify how the data should be processed. For example, you can request that the Connection check for a minimum input record size, strip quotation marks from input text or add quotation marks to output text, and skip a specific amount of data before beginning processing. More information on using each source and target option follows.

**Note** Source Options are used when an Activity references the Text Connection as its Source Connection. Target Options are used when an Activity references the Text Connection as its Target Connection.


New Text Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

Source Options

| | |
|---|---|
| Input Options: | ☐ Check for Input Values Within Quotes |
| Minimum Record Size: | 1 bytes |
| Maximum Record Size: | 128000 bytes |
| Skip Records: | Skip first 0 records. |
| Skip Bytes: | Skip first 0 bytes. |

Target Options

☐ Output Values Within Quotes
☐ Append Data to Output File
Fill Character/String:

This is the protected text area of the form.

[None] You have new mail. Office

### Source Options

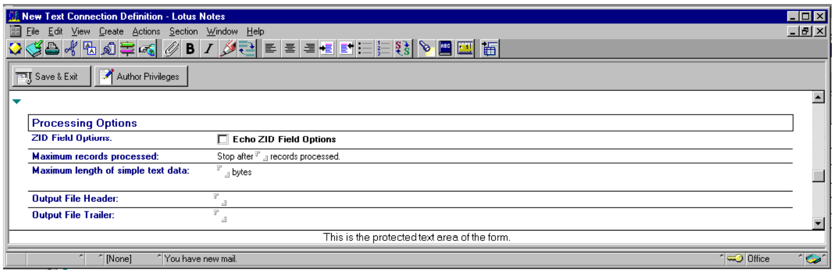
| *Field* | *Description* |
|---|---|
| Check for Input Values Within Quotes | Select this option if you want Text Connector to remove quotation marks that surround input text. If you do not select this option, Text Connector puts text into the document fields exactly as supplied, so surrounding quotes would go into the document with the text data. |
| Minimum Record Size | Specify the minimum record size. Use this option when you need to set a minimum input record length. The default value is 0, which indicates that no minimum record length exists. Specifying a value greater than 0 forces Text Connector to bypass input records that are smaller than the specified minimum. This feature is very useful for bypassing junk records or blank lines that might be present in an input text file. |
| Skip first ‘n’ records | Specify the number of records Text Connector should skip before it begins processing. This is useful for ignoring header information or extraneous white space at the beginning of a data file. |
| Skip first ‘n’ bytes | Specify the number of bytes the Connector should skip before it begins processing. This is useful for ignoring header information or extraneous whitecap at the beginning of a data file. |

## Target Options

| *Field* | *Description* |
|---|---|
| Output Values Within Quotes | Select this option if you want to output your data with surrounding quotes. If you do not select this option, Text Connector outputs the data exactly as it exists. |
| Append data to output file | Select this option if you want to append the data to an existing file. If you do not select this option, Text Connector processes output text as a new file and overwrites any existing data in the file. If you elect to append the data to an existing file, Text Connector writes the string specified by the record delimiter to the output file before any document processing begins. If the file does not exist, Text Connector creates one. |
| Fill character/string | Specify the character that you want to occupy unused or whitecap areas in output records. If you do not enter a value, Text Connector defaults to ASCII blanks. |

## Processing Options

The Text Connector offers processing options for both source and destination forms. The fields and options are the same for both source and destination. These fields are described in the table below.


New Text Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

Processing Options

| | |
|---|---|
| ZID Field Options. | ☐ Echo ZID Field Options |
| Maximum records processed: | Stop after ⌜ ⌟ records processed. |
| Maximum length of simple text data: | ⌜ ⌟ bytes |
| Output File Header: | ⌜ ⌟ |
| Output File Trailer: | ⌜ ⌟ |

This is the protected text area of the form.

[None] You have new mail. Office

| *Field* | *Description* |
|---|---|
| ZID Field Options | Echo ZID Field Options – You can select the option to echo ZID field options. ECHO is a useful parameter for debugging your ZID scripts. Basically, specifying ECHO causes LEI to display each line of your ZID script as an LEI log entry as it is processed. Any error messages are also listed with the associated line number that generated the error. |
| Maximum records processed | Stop after … records processed – You can specify the maximum number of records Text Connector should process. Enter the number in the brackets. If you do not specify a number, Text Connector processes all records. |
| Maximum length of simple text data | bytes – Signals LEI to use this length as the maximum for Text type fields. This option is often used when a metadata is being created in a relational database and text lengths and types are important. The default text length is 32Kb. |

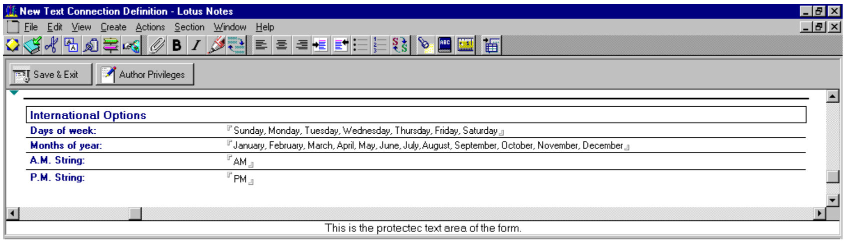
The following options are for target Connections only.

| *Field* | *Description* |
|---|---|
| Output File Header | Specify the text to be added to the output file before the first record. (This may be useful when generating a table for a report and you wish to add comments or description before the table. It may also be used to start an HTML page for a web site which will contain a table of data.) |
| Output File Trailer | Specify the text to be added to the output file after the last record. (This may be useful when generating a table for a report and you wish to add a key, comments, or a description after the table. It may also be used to complete a generated HTML page which was started with the Output File Header option.)<br>The combination of formatting in the ZID and the use of the Output File Header and Trailer may be used to generate an HTML page for a web site. The output description includes the HTML table row start mark with each field being surrounded by the start and stop marks for the column and concluding with the row end mark, as follows:<br>**`(row): TYPE TEXT VALUE "<tr><td>"`**<br>**`Idno: TYPE TEXT UNTIL "</td><td>"`**<br>**`Name: TYPE TEXT UNTIL "</td><td>"`**<br>**`Tel:  TYPE TEXT UNTIL "</td><td>"`**<br>**`Balance: TYPE NUMBER.2 UNTIL "</td><td>"`**<br>**`Payment: TYPE NUMBER.2 UNTIL "</td><td>"`**<br>**`Date: TYPE DATE "</td>" UNTIL FORMAT MM-DD-YYYY`**<br>**`(endrow):  TYPE TEXT VALUE "</tr>"`**<br>The Output File Header starts the HTML page.<br>**`  <html>`**<br>**`  <head><title>Generated Page</title></head>`**<br>**`  <body>`**<br>**`  <center><h1>This is a generated page</h1></center>`**<br>**`  <table>`**<br>**`  <tr>`**<br>**`  <th>ID</th>`**<br>**`  <th>Name</th>`**<br>**`  <th>Telephone</th>`**<br>**`  <th>Balance</th>`**<br>**`  <th>Payment</th>`**<br>**`  <th>Date</th>`**<br>**`  </tr>`**<br>The Output File Trailer completes the HTML page.<br>**`</table>`**<br>**`  </body>`**<br>**`  </html>`** |

## International Options

The Text Connector enables you to process date information using text versions of days of the week and months of the year, and A.M. and P.M. abbreviations. By default, the days of the week, month names, and meridian indicators are in English. You can change the English translations to other languages by setting them in International Options. Enter the notation you want for each day, month, or meridian indicator in the place of the English word. Translation is based on placement in the list, so you must include all seven days of the week and all twelve months.

The international options work with the FORMAT options.


New Text Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
This is the protectec text area of the form.

International Options

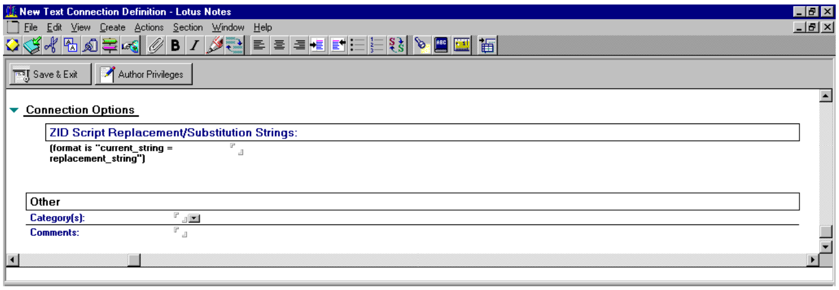
| | |
|---|---|
| Days of week: | Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday |
| Months of year: | January, February, March, April, May, June, July, August, September, October, November, December |
| A.M. String: | AM |
| P.M. String: | PM |

| *Field* | *Description* |
|---|---|
| Days of week | Modify the list to use language specific days of the week. |
| Months of year | Modify the list to use language specific months of the year. |
| AM/PM | The A.M. and P.M. strings can be specified. You can change the English defaults to foreign languages using the International Options. |

The FORMAT syntax for a date is FORMAT MMM_dd_YYYY where MMM is the input text month name. When you select International Options, Text Connector replaces MMM with the appropriate month as you specify in the International Options list.

## Connection Options

The Text Connector enables you to specify a substitute string.


New Text Connection Definition - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges

Connection Options

ZID Script Replacement/Substitution Strings:

(format is "current_string = replacement_string")

Other

Category(s):

Comments:

| *Field* | *Description* |
|---|---|
| ZID File Replacement/ Substitution Strings | This option enables you to substitute a string for the current string in the Text Connector. |

## Text Data Types

The following tables show the correspondences of Text Connector data types to LEI data types for various types of Activity.

- (p) – Indicates that if Allow Precision Loss is not enabled, then an error will be generated on the type match. Allowing precision loss is the default.
- (o) – Indicates that overflow checking will be performed when data is being transferred. If an overflow occurs and Truncate Data When Necessary is enabled, then the data is truncated; if not an error is generated.
- Text, Number, or Time indicates single-value unless otherwise specified.
- The Text Connector option Maximum Length for Text Data alters the default bound of 64996 for LEI Text.

For more information on the organization of these tables, see the section

## Execute

| *Text* | *LEI* |
|---|---|
| Number (suffix .n for implied decimal) | single-value |
| Number (with defined SEPARATOR string) | allow multi-values |
| DateTime | single-value |
| Datetime (with defined SEPARATOR string) | allow multi-values |
| Text | single-value |
| Text (with defined SEPARATOR string) | allow multi-values |
| Rich Text | |
| Numeric | |
| Currency | |

## Fetch

| *LEI* | | *Text* |
|---|---|---|
| Int | | Number (p, 0) |
| Float | | Number |
| Currency | | Currency |
| Numeric | | Numeric |
| Datetime | | Datetime |
| Text (0) | | Text, Rich Text, Formula |
| Binary (0) | BLOB | Text, Number allow multiple values, Time allow multiple values, Rich Text |
| | composite | Text (single-value), Rich Text |
| | number list | Number, Text (single-value) |
| | datetime list | Time, Text (single-value) |
| | text list | Text |

## Insert

| *LEI* | | *Text* |
|---|---|---|
| Int | | Number |
| Float | | Number |
| Currency | | Currency |
| Numeric | | Numeric |
| Datetime | | Datetime |
| Text | | Text, Rich Text |
| Binary | BLOB | Text (o), Number (allow multiple values) (o), Time (allow multiple values) (o), Rich Text |
| | composite | Text (single-value) (o), Rich Text |
| | number list | Number (o), Text (single-value) (o) |
| | datetime list | Time (o), Text (single-value) (o) |
| | text list | Text (o) |

## Create

| *LEI* | | *Text* |
|---|---|---|
| Int | | Number (single-value) |
| Float | | Number (single-value) |
| Currency | | Currency |
| Numeric | | Numeric |
| Datetime | | Time (single-value) |
| Text | bounded | Text (single-value) |
| | unbounded | Rich Text |
| Binary | BLOB | Text (single-value) |
| | composite | Rich Text |
| | number list | Number (allow multiple values) |
| | datetime list | Time (allow multiple values) |
| | text list | Text (allow multiple values) |

# Chapter 16 MetaConnection

This chapter provides an introduction to LEI MetaConnectors and contains information for creating and using the different types of MetaConnectors.

## Introduction to MetaConnectors

A MetaConnector is a special kind of LEI Connector that provides preprocessing operations on Connector data prior to transfer within a defined Activity form.

LEI supports the following MetaConnectors:

- Collapse/Expand MetaConnector – The Collapse/Expand MetaConnector provides the capability to take multiple records from a data source table, and "Collapse" them to a single form field, and perform the reverse operation, or "Expand" the data into multiple records.
- Connection Broker MetaConnector – The Connection Broker MetaConnector allows data to be sent to a Connector using different user credentials, or to different Connectors within a single Activity.
- Metering MetaConnector – The Metering MetaConnector provides a way to collect statistical usage data. A Metering MetaConnection can identify and quantify data access.
- Order MetaConnector – The Order MetaConnector is useful when ordering data sets from different server sources. For example, a DB2 table on an AS400 system and a Notes database on a Domino server may use different order systems when ordering data. This can result in a data set comparison problem when using a LEI Replication Activity, which requires data sets to be ordered in parallel for data set comparisons. The Order MetaConnector may be applied in this situation to preprocess the data sets to be compared during the Activity, ensuring the order pattern will be in parallel for accurate data set comparisons during the Replication Activity.

- Trace MetaConnector – The Trace MetaConnector enables you to record Connector trace data. Options exist for specifying a time stamp, where to capture trace data, and what file name to use for logging output. If there is a problem with your Activity, the Trace MetaConnector is a tool for you to use when troubleshooting. Also, trace output is useful for Lotus support staff when troubleshooting customer calls.

Once a MetaConnector is created, the MetaConnection document can be found in the views under Connections.

## Collapse/Expand MetaConnector

Use a Collapse/Expand MetaConnector to group or ungroup multiple records, with a common key value, into a single record with multi-value fields. The Collapse/Expand MetaConnector wraps another Connector or MetaConnector and collapses records on Fetch operations. It then expands them on write operations according to user-defined criteria. The Collapse/Expand MetaConnector has no effect on metadata operations – it simply passes through Create, Drop, and Action statements.

Since multiple records are combined into or generated from one record, Writeback operations are not available. Writeback operations are simulated with keyed operations. When performing a Writeback operation through a Collapse/Expand MetaConnector, make sure to provide the relevant key values.

In all cases, a key must be defined as one or more fields. When producing a result set or fetching data, all fields that are not part of the key are retrieved as multi-value datatypes (binary stream formats text list, number list, and datetime list). There are no multi-value equivalents to binary datatypes. On retrieval of data, records are fetched until the key value changes, and then all values for each field are combined into multi-value data objects. When writing data, the inverse occurs – multiple write operations are invoked (using an optional additional key field or fields for keyed operations), with the key fields staying constant, and the multi-value fields changing on each operation.

It is crucial to the proper behavior of a Collapse/Expand MetaConnection to ensure that any external result set is properly ordered by the grouping keys. This ordering is accomplished by also using the Order MetaConnector. All records with the same grouping key field values are combined (collapsed) until a different key value is encountered, so if all the equivalent keys are not returned consecutively, the corresponding records will not all be combined.

**Note** One of the primary uses of a Collapse/Expand MetaConnection is to allow a single document in a Notes database to relate to multiple records in an external relational database. Although there are no restrictions that Notes or a relational database be used with this MetaConnector, Notes fully supports multi-value fields, and this is the most common scenario.

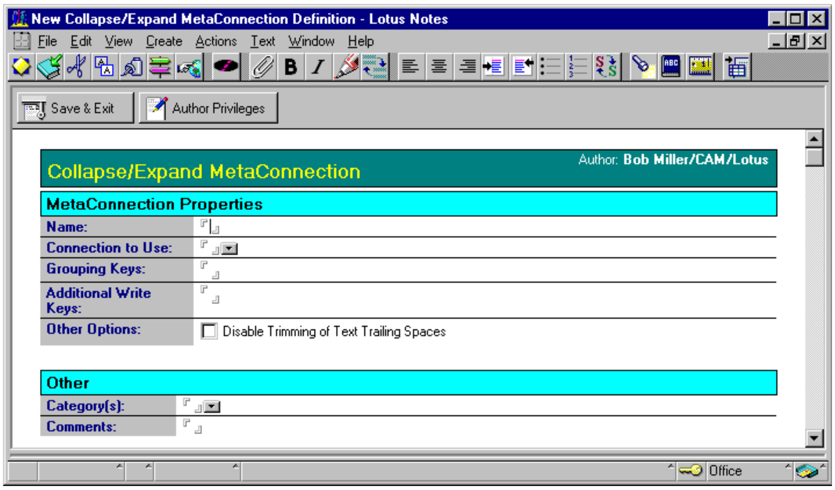
## Connectivity to a Collapse/Expand MetaConnection

There are no connectivity requirements beyond those for the underlying Connection to which the Collapse/Expand MetaConnection is connecting. Since there is no external database for the Collapse/Expand MetaConnection itself, there is no connectivity test.

## Collapse/Expand MetaConnection Document

Click the Create Connector icon in the Navigator or choose Create – MetaConnection – Collapse/Expand to access the Connection document. Use this document to define a Collapse/Expand MetaConnection. The fields and options in this document are described below.

As with Connectors, you can create as many instances of the form as you want, each with different parameters. You can define multiple Order MetaConnections by saving them with different Collapse/Expand MetaConnection Names. You can also configure different Collapse/Expand MetaConnections for the same Connector, and sort the Connector differently depending on the Collapse/Expand MetaConnection chosen for use with an Activity.



Collapse/Expand MetaConnection

Author: Bob Miller/CAM/Lotus

MetaConnection Properties

| | |
|---|---|
| Name: | |
| Connection to Use: | |
| Grouping Keys: | |
| Additional Write Keys: | |
| Other Options: | ☐ Disable Trimming of Text Trailing Spaces |

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

## Collapse/Expand MetaConnector Properties

| *Field* | *Description* |
|---|---|
| Name | The name that identifies the Connector. |
| Connection to Use | The name of the underlying Connection which is providing the data. This will often be an Order MetaConnector. |
| Grouping Keys | The field or list of fields that define the grouping key. One internal record is constructed from all external records with the same key value. To ensure that all records with this key value are collapsed into a single multi-value record, make sure that the external result set is ordered using the Order MetaConnector by the grouping key fields. |
| Additional Write Keys | Additional keyfield or fields which define the unique key within a collapsed record. For example, if four orders by a single customer are collapsed into one document, with the customer number as the key, and one order is changed, then the order number may be required to ensure the key's uniqueness across individual external records for this customer on expansion. |
| Disable Trimming of Text Trailing Spaces | In normal operation, whenever text data is fetched from a Connection, trailing spaces are removed from the text. The property NoTrimText has been added to the Direct Transfer and Replication Activities. It disables the trimming of trailing spaces in text data. When using this option in a Direct Transfer or Replication Activity that uses a Collapse/Expand MetaConnection, this option must also be selected here in the MetaConnection definition. |

## Using the Collapse/Expand MetaConnector with the Replication Activity

When using a Collapse/Expand MetaConnection with a Replication Activity, the key fields and the grouping keys need to be the same and in the same order in the MetaConnection definition as in the Replication Activity definition to work properly.

## Collapse/Expand MetaConnector Data Types

Since the Collapse/Expand MetaConnector has no external database, it always operates with LEI datatypes. For key fields, all LEI datatypes convert within their class (number, datetime, or stream); in addition, precision and overflow checking are performed as needed. For non-key fields, on Fetch, external single-value data types map to internal multi-value datatypes, and vice-versa on output. Therefore, an external number class type maps to a number list; a datetime to a datetime list; and text to a text list. Note that binary data types (unformatted BLOB or formatted binary types such as composite) cannot be mapped to multi-value types.

For Create operations, see the underlying Connector documentation.

## Connection Broker MetaConnector

This MetaConnector allows data to be directed to/from different Connections, based on contact information that is supplied with each row of data. Using this MetaConnector within an Activity form adds in support for end-user authentication to external-connector data for subsequent data write or update operations to Connector-defined data.

The additional Connection properties are specified through additional fields in the field list supplied by the client, for example, as entered by an end user in a Domino form. The MetaConnector properties allow the Connector to accept individual username and password parameters submitted, and uses these credentials instead of the username and password defined to a specified Connector Document to authenticate individual user accounts to the Connector source.

By using the Connection Broker MetaConnector within an Activity definition, an Activity can use the Connection Broker to supply user credentials other than what are specified in the underlying default Connector when writing to the Connector. These credentials are supplied in fields input by an end user to a Domino Form, sent through the Connection Broker to the enterprise system during Activity processing. This option is most useful when using the RealTime Notes Activity.

The Connection Broker used the default Connection (as defined to the Connector document) as a starting point. Each row of data is examined for the fields specified in the form. If data is found in these fields, they will override that component of the default Connection to create a new Connection.

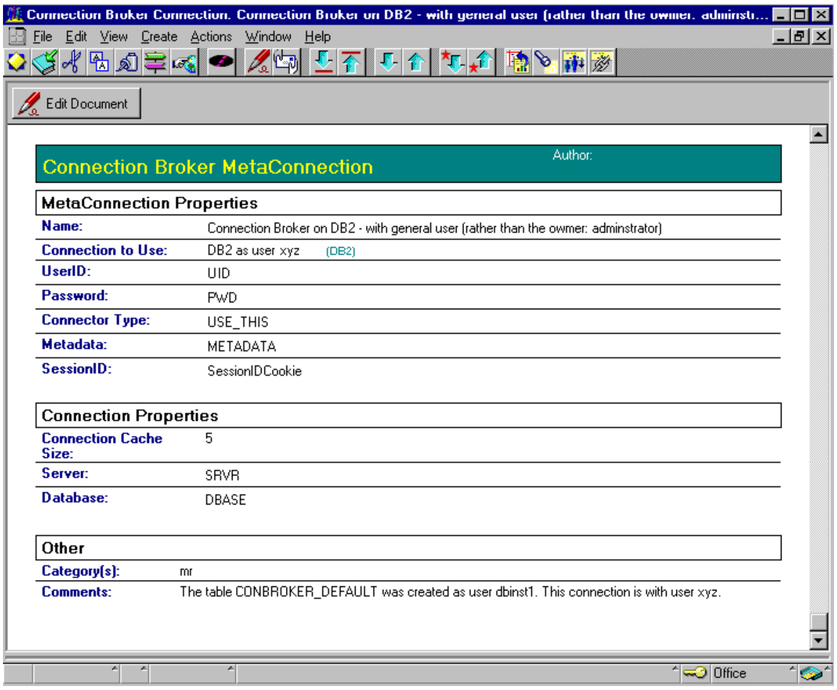
Typically, the user credentials (UserID and Password) will be modified and input by an end user client to a Domino application for transfer to the external Connector server, but it is also possible to modify the Connector type, database name, server name, and metadata name.

As the Connection Broker looks in the data passed into it for the information, it is generally sent to the destination defined in an Activity.

If you attempt to truncate data through a Connection Broker MetaConnector by selecting either an Activity option such as Direct Transfer – Overwrite Existing Data or the LSX LCConnection.Action method TRUNCATE, only records in currently open Connections are truncated. Since a Direct Transfer Activity performs the overwrite operation before inserting any records, it does not have any effect when it is done through the MetaConnector.

## Connection Broker MetaConnection Document

The Connection Broker MetaConnection document is shown below. Click the Create Connection icon in the navigator and select Connection Broker or choose Create – MetaConnection – Connection Broker to access the document. The fields and options in the document are described below.


Connection Broker Connection. Connection Broker on DB2 - with general user (rather than the owner. adminstr...
File Edit View Create Actions Window Help
Edit Document
Office

Connection Broker MetaConnection

Author:

**MetaConnection Properties**

| | |
|---|---|
| Name: | Connection Broker on DB2 - with general user (rather than the owmer: adminstrator) |
| Connection to Use: | DB2 as user xyz (DB2) |
| UserID: | UID |
| Password: | PWD |
| Connector Type: | USE_THIS |
| Metadata: | METADATA |
| SessionID: | SessionIDCookie |

**Connection Properties**

| | |
|---|---|
| Connection Cache Size: | 5 |
| Server: | SRVR |
| Database: | DBASE |

**Other**

| | |
|---|---|
| Category(s): | mr |
| Comments: | The table CONBROKER_DEFAULT was created as user dbinst1. This connection is with user xyz. |

## Connection Broker MetaConnection Properties

The contact information is specified through Connection Broker MetaConnection properties that specify the field names that will be used to communicate the different pieces of contact information. Supported properties include the following:

| *Field* | *Enter* |
|---|---|
| Name | A user created name for this Connection. |
| Connection to Use | Name of a Connection the user previously created. This Connection will be the default. |
| Connector Type | The column or field name that will alert the Connection Broker as to the Connection properties to look for. Values entered in this field include: Oracle, Sybase, DB2, ODBC, Text, File, and so on. |
| UserID | The column or field name that contains the user ID or user name. |
| Password | The column or field name that contains the password. |
| Metadata | The column or field that contains the table or form name. |
| SessionID | The MetaConnector will fill this field in with a session “cookie” to differentiate between concurrent requests. This field and value must be supplied with subsequent calls associated with the original request. The field will be created and added to the fieldlist by the MetaConnector if it does not already exist in the supplied fieldlist. A default name for this new field is supplied or it can be altered if there is a conflict with data. |
| Connection Cache Size | The MetaConnector will maintain a number of open Connections to improve speed for repeat calls. The number of Connections that are maintained is set with this property. |
| Server | Enter name of server if required by Connector Type. |
| Database | Enter name of database if required by Connector Type. |

Each of these specify the name of the field used to modify the associated Connection property. These are the standard properties for a Connection. You can specify nonstandard properties using the Connection name and values properties.

When the “overwrite existing data” option is selected in the Activity document, only the data in the default Connection is affected.

If any of the field name properties are not set, or if there is no data in one or more of the specified fields for a particular row of data, default contact information is used. The base Connector (the TOKEN_CONNECTOR property) is used for default connect information.

If there are no field name properties set, or if there is no data in the specified fields, the MetaConnector simply operates as a pass through, using the default underlying Connector document user and password settings.

## Setting up a RealTime Activity Using the Connection Broker

In this example, we will first use a Direct Transfer Activity to create a DB2 table. We will allow all users to read this table. However, only users with the table owners username and password will be able to change the data. Then we will create a RealTime Activity using the Connection Broker to update the DB2 table, but, only users with the administrator username and password will be able to update data.

Although any database that can be used with a RealTime Activity is appropriate for this example, we will use the data contained in empsamp.nsf, (a sample Notes database that ships with LEI) to create the test data. By setting up the example in this way, any relational database can be used as the backend.

### Create Backend Table

1. Using the emps form of the empsamp.nsf database as the source, create a Direct Transfer Activity to DB2 and create a table called CONBROKER_DEFAULT (see Chapter 23, Direct Transfer Activity and Appendix E, Tutorials). Create the target DB2 Connector as the owner of the table so that the table gets created in this user's name (that is, the owner's name).
2. To this newly created CONBROKER_DEFAULT table, you now need to add the columns that will contain the Connection parameters; to accomplish this task, use the following SQL or database tools:

   ```
   ALTER TABLE <table owner>.CONBROKER_DEFAULT

   ADD COLUMN USE_THIS VarChar (12)

   ADD COLUMN USERID VarChar (15)

   ADD COLUMN PWD VarChar (12)

   ADD COLUMN METADATA VarChar (20)

   ADD COLUMN SRVR VarChar (12)

   ADD COLUMN DBASE VarChar (12)
   ```

   **Caution** Avoid using column names, such as password and user, that could be interpreted by different databases as key words.
3. Using SQL or the database tools, restrict the "world" to have read-only (select) access.

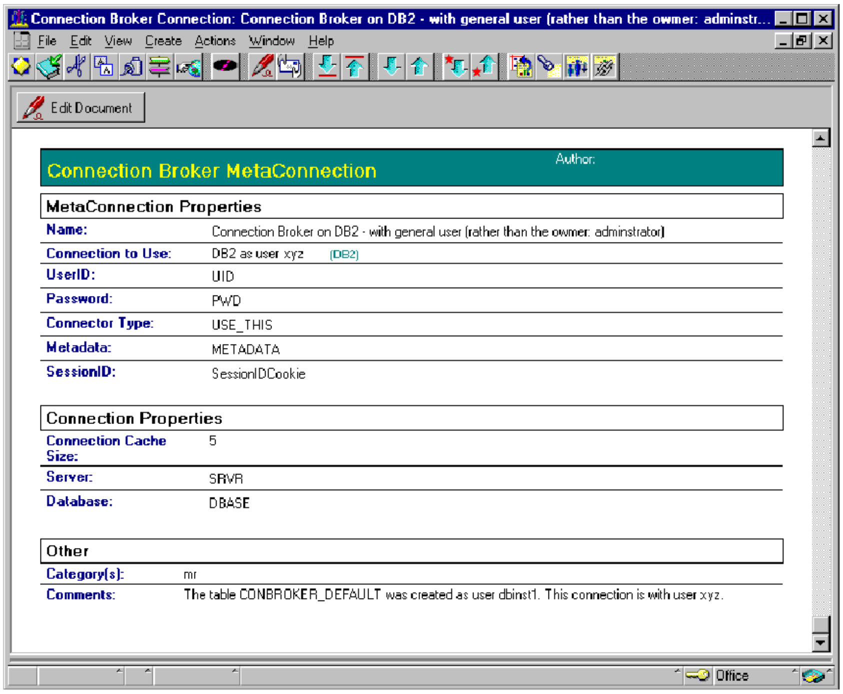
   ```
   GRANT SELECT ON TABLE CONBROKER_DEFAULT TO GROUP WORLD
   ```

### Creating a Notes Form

1. Create a shortcut to open the empsamp.nsf database in Domino Designer and copy the form entitled "emps". Paste this form into the Notes database that you want to use as the front end to the RealTime Activity. You need to add the fields that will contain the Connection parameters to this form. For this example that would mean fields to correspond to the following backend columns: USE_THIS, USERID, PWD, METADATA, SRVR and DBASE. For simplicity, we will use the same names as contained in the backend table.
2. Create a Notes view for this form.

## Creating a Connection Broker MetaConnector

1. Create a new DB2 Connector with a general user's username and password.
2. Set up the Connection Broker metaconnector form to sit on top of the DB2 Connector form you have just created in Step 1. This DB2 Connector will be identified in the "Connection Name" field of the Connection Broker form. Remember that the values that go in these fields are the column names in the backend default table.


Connection Broker Connection: Connection Broker on DB2 - with general user (rather than the owner: adminstr...
File Edit View Create Actions Window Help
Edit Document
Office

Connection Broker MetaConnection

Author:

MetaConnection Properties

| | |
|---|---|
| Name: | Connection Broker on DB2 - with general user (rather than the owner: adminstrator) |
| Connection to Use: | DB2 as user xyz (DB2) |
| UserID: | UID |
| Password: | PWD |
| Connector Type: | USE_THIS |
| Metadata: | METADATA |
| SessionID: | SessionIDCookie |

Connection Properties

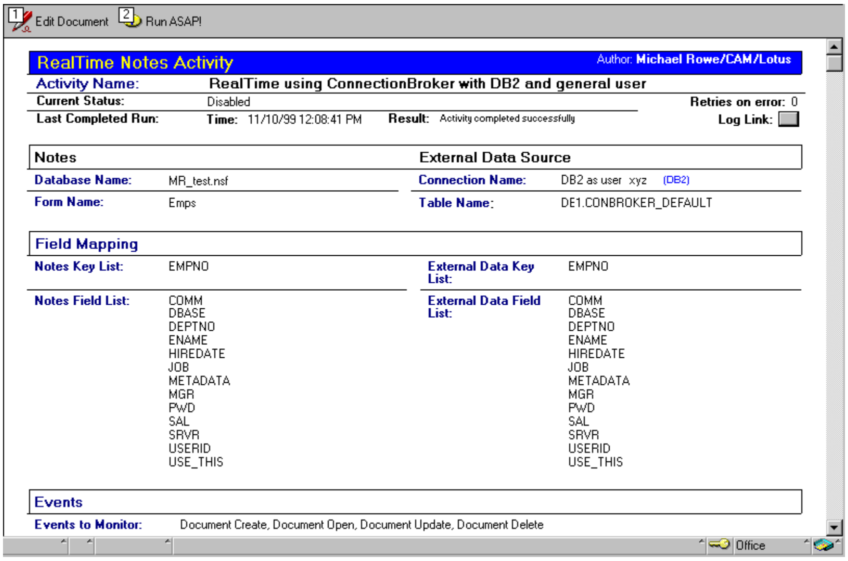
| | |
|---|---|
| Connection Cache Size: | 5 |
| Server: | SRVR |
| Database: | DBASE |

Other

| | |
|---|---|
| Category(s): | mr |
| Comments: | The table CONBROKER_DEFAULT was created as user dbinst1. This connection is with user xyz. |

## Creating the RealTime Activity

Next you create the RealTime Activity. See the illustration below and Chapter 25, RealTime Notes Activity, and Appendix E, Tutorials, for more information on creating a RealTime Activity.



RealTime Notes Activity — Author: Michael Rowe/CAM/Lotus

| | |
|---|---|
| Activity Name: | RealTime using ConnectionBroker with DB2 and general user |
| Current Status: | Disabled |
| Retries on error: | 0 |
| Last Completed Run: | Time: 11/10/99 12:08:41 PM Result: Activity completed successfully |
| Log Link: | |

Notes

| | |
|---|---|
| Database Name: | MR_test.nsf |
| Form Name: | Emps |

External Data Source

| | |
|---|---|
| Connection Name: | DB2 as user xyz (DB2) |
| Table Name: | DE1.CONBROKER_DEFAULT |

Field Mapping

| | | | |
|---|---|---|---|
| Notes Key List: | EMPNO | External Data Key List: | EMPNO |
| Notes Field List: | COMM<br>DBASE<br>DEPTNO<br>ENAME<br>HIREDATE<br>JOB<br>METADATA<br>MGR<br>PWD<br>SAL<br>SRVR<br>USERID<br>USE_THIS | External Data Field List: | COMM<br>DBASE<br>DEPTNO<br>ENAME<br>HIREDATE<br>JOB<br>METADATA<br>MGR<br>PWD<br>SAL<br>SRVR<br>USERID<br>USE_THIS |

Events

| | |
|---|---|
| Events to Monitor: | Document Create, Document Open, Document Update, Document Delete |

## Initializing Keys in the RealTime Activity

In LEI, a simple Direct Transfer Activity will create "stub" documents a result of initializing keys. You will need these "stub" documents for the RealTime Activity.

1. Create a Direct Transfer Activity using the already created DB2 Connector and the following SQL statement:

```
SELECT EMPS FROM <table owner>.CONBROKER_DEFAULT
```

## Running the Activity

1. From the LEI navigator, start the Activity.
2. Open the Notes front end database and make sure that the stub document is there and has the data fields populated.

From this point, only the user with the correct authority can alter the backend data by entering the table owner's username and password. All other users (depending on table constraints) will only be able to read the backend records.

## Meter MetaConnector

Use the Meter MetaConnector to track data flowing to and from another Connection or MetaConnection. The Meter MetaConnector has no effect on any operations; it simply monitors the data transferred.

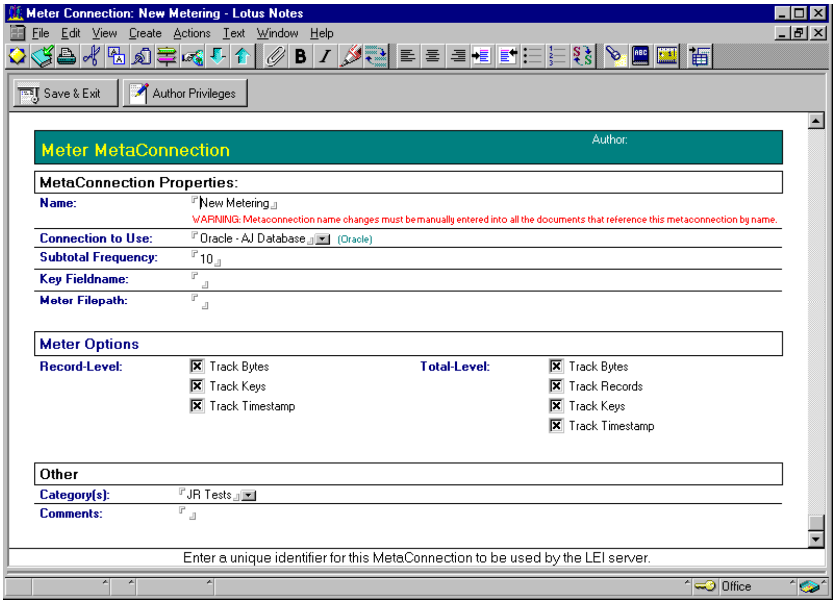
Results from the metering are logged to a file. The specific information logged is configurable through use of the Meter MetaConnection document.

## Connectivity to a Meter MetaConnection

There are no connectivity requirements beyond those required for the underlying Connection to which the Meter MetaConnection is connecting. Since there is no external database for the Meter MetaConnection itself, there is no connectivity test.

## Meter MetaConnection Document

The Meter MetaConnection document is shown below. Choose Create – MetaConnection – Meter, or click Create Connection and select Meter from the LEI Administrator to access the Meter MetaConnection document. The fields and options in the document are described in the table below.



Meter MetaConnection
Author:

MetaConnection Properties:

| | |
|---|---|
| Name: | New Metering<br>WARNING: Metaconnection name changes must be manually entered into all the documents that reference this metaconnection by name. |
| Connection to Use: | Oracle - AJ Database (Oracle) |
| Subtotal Frequency: | 10 |
| Key Fieldname: | |
| Meter Filepath: | |

Meter Options

| | | | |
|---|---|---|---|
| Record-Level: | [✓] Track Bytes | Total-Level: | [✓] Track Bytes |
| | [✓] Track Keys | | [✓] Track Records |
| | [✓] Track Timestamp | | [✓] Track Keys |
| | | | [✓] Track Timestamp |

Other

| | |
|---|---|
| Category(s): | JR Tests |
| Comments: | |

Enter a unique identifier for this MetaConnection to be used by the LEI server.

## Meter MetaConnection Properties

| *Field* | *Description* |
|---|---|
| Name | The name that identifies the Connection. |
| Connection to Use | The name of the underlying Connection which is providing the data. |
| Subtotal Frequency | How often to record subtotal results computed in number of records. Subtotals can be printed out during the process. The number specified is how many records to process to get subtotal information. If you process 1000 records, and the subtotal frequency was set to 300, you'd get 3 subtotals and then a final tally. |
| Key Fieldname | Optional field containing the value to group results by. Use this to generate results grouped by a particular data field value. |
| Meter Filepath | Filepath of the meter log file.<br><br>**Note** The meter log file is written to a directory accessible from the LEI Server. To transfer this file into a database, follow the metering Activity with a direct transfer using the File system Connector to access the meter log file, and any other Connector as the destination. |

## Meter MetaConnector Options

Meter MetaConnector options are listed in the following table.

| *Field* | *Description* |
|---|---|
| Record Level | Select the items to track at the record level – bytes transferred, records transferred, key values, Timestamp. |
| Total Level | Select the items to track at the subtotals and totals level – bytes transferred, records transferred, key values, time stamp. |

## Meter MetaConnector Supported Characteristics

Supported characteristics for the Meter MetaConnector are listed below. For a definition of these terms, see Chapter 5, Introduction to Connectors, of this manual.

| | |
|---|---|
| Connection Properties | As supported by the underlying Connector. |
| Standard Functionality | Full, as supported by the underlying Connector. |
| Writeback Support | Full, as supported by the underlying Connector. |
| Writeback Index | N/A |
| Statement Syntax | See the underlying Connector documentation. |
| Condition Syntax | See the underlying Connector documentation. |
| Connection Properties | Connector, LogRecord, LogSubtotal, LogTotal, SubtotalFrequency, KeyFieldname, MeterFilename, plus those supported by the underlying Connector. |
| Array Transfer | See the underlying Connector documentation. |
| Actions Supported | See the underlying Connector documentation. |
| Catalog Types | See the underlying Connector documentation. |
| Create Types | See the underlying Connector documentation. |
| Drop Types | See the underlying Connector documentation. |

## Meter MetaConnector Data Types

Since the Meter MetaConnector has no external database, it always operates with LEI datatypes. All LEI datatypes convert within their class (number, datetime, or stream). Precision and overflow checking are performed as needed. For more information, see the underlying Connector documentation.

## Order MetaConnector

Use an Order MetaConnection to obtain consistent ordering of result sets from other Connections The Order MetaConnector wraps another Connector and orders the result set based on user-defined criteria. Order options are available. The Order MetaConnector has no effect on data operations, other than producing a result set or fetching – it simply passes through Insert, Update, Remove, Create, Drop, and Action statements.

The Order MetaConnector loads and sorts the entire result set produced by its subconnector (through an Execute, Select, or Catalog operation) and then returns records in order for Fetch operations. Since the entire result set is loaded into memory, Writeback operations are not available. Writeback operations are simulated with keyed operations. When performing a Writeback operation through an Order MetaConnection, make sure to provide the relevant key values).

One of the primary problems solved by the Order MetaConnector is that of different sort orders during replication. For example, when replicating between Oracle and Notes, there are two potential sort order problems. First, the two databases handle case sensitivity differently. Second, certain punctuation characters are sorted differently. These differences may appear between any disparate databases, depending on the options selected for that database and the data set being replicated. For more information on database sort orders, contact your database system administrator.

Sort order differences may result in data that is out of sync. This results in numerous unnecessary insertions and deletions. While the end result is correct, the additional change operations can seriously degrade performance or cause problems with triggers and other events. To correct this problem, use an Order MetaConnection on both sides of the replication. Choose the same Order MetaConnection sorting options for each Connection. This will replace the external database’s sorting with its own consistent sorting . In addition, using the same character set for both sides of the replication may increase efficiency.

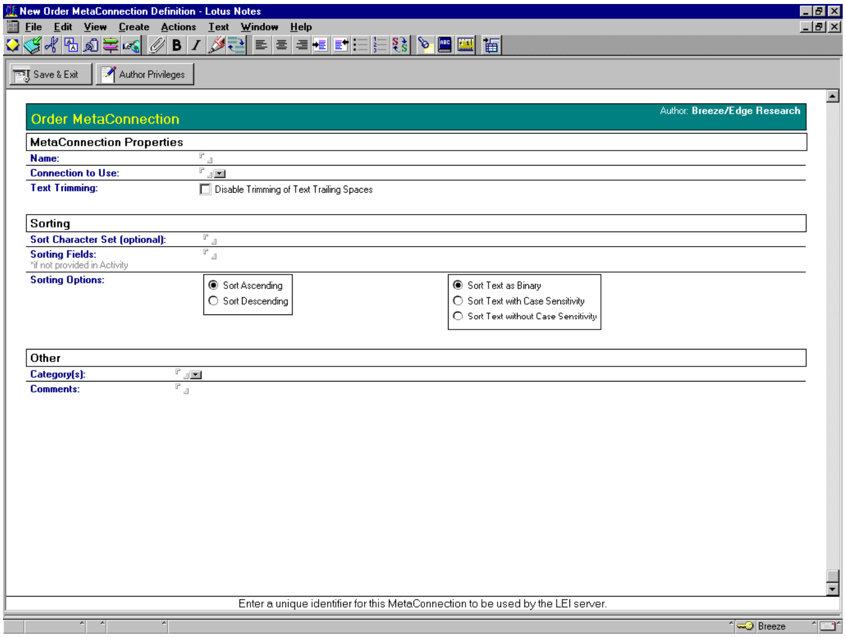
## Connectivity to an Order MetaConnection

There are no additional requirements for connectivity beyond those of the underlying Connection to which the Order MetaConnection is connecting. Also, since there is no external database for the Order MetaConnection itself, there is no connectivity test.

## Order MetaConnection Document

Choose Create – MetaConnection – Order, or click Create Connection in the Navigator and select Order to access the Order MetaConnection document, shown below. Use this document to define an Order MetaConnection. The fields and options in this document are described below.

You can define multiple Order MetaConnections by saving them with different Order MetaConnection Names. You can also configure different Order MetaConnections for the same Connection, and sort the Connection differently depending on the Order MetaConnection chosen for use with an Activity.


New Order MetaConnection Definition - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Breeze

Order MetaConnection

Author: Breeze/Edge Research

MetaConnection Properties

Name:

Connection to Use:

Text Trimming: ☐ Disable Trimming of Text Trailing Spaces

Sorting

Sort Character Set (optional):

Sorting Fields:
*if not provided in Activity

Sorting Options: [✓ Sort Ascending] Sort Descending; [✓ Sort Text as Binary] Sort Text with Case Sensitivity Sort Text without Case Sensitivity

Other

Category(s):

Comments:

Enter a unique identifier for this MetaConnection to be used by the LEI server.

| *Field* | *Description* |
|---|---|
| Name | The name that identifies the MetaConnection. |
| Connection to Use | The name of the underlying Connection which is providing the data. |
| Sort Character Set (optional) | Optional character set to force text data into. By default, the underlying Connection’s character set is used. To provide a character set, use the corresponding constant suffix (character sets are listed in Appendix C). For example, for character set Code Page 932, represented by the LEI constant LCSTREAMFMT_CP932, enter “CP932” here. |
| Sorting Options | Whether to sort in ascending (the default) or descending order. Replication depends on an ascending sorting to operate properly, therefore do not use the descending option with a Replication Activity.<br>Text sort order – binary (the default and most efficient), case-sensitive, or case-insensitive. |
| Sorting Fields | Enter a list of fields in which to sort data. For Activities where ordering is specified, such as Replication, these selections are overridden and therefore ignored in the Connector form. But for other Activities, this is a place to provide the ordering (which was not available before). |
| Disable Trimming of Text Trailing Spaces | In normal operation, whenever text data is fetched from a Connection, trailing spaces are removed from the text. The property NoTrimText has been added to the Direct Transfer and Replication Activities. It disables the trimming of trailing spaces in text data. When using this option in a Direct Transfer or Replication Activity that uses an Order MetaConnection, this option must also be selected here in the MetaConnection definition. |

## Using the Order MetaConnector with a Replication Activity

Replication depends on an ascending sorting to operate properly. Do not use the Sort Descending option of this MetaConnector when performing Replication Activities.

## Order MetaConnector Data Types

Since the Order MetaConnector has no external database, it always operates with LEI data types. All LEI data types convert within their class (number, datetime, or stream). Precision and overflow checking are performed as needed. For operations other than Result Set Production and Fetch, see the underlying Connector documentation.

## Trace MetaConnector

The Trace MetaConnector allows you to trace events associated with a specified sub connection. You may specify options including where to capture data (either to the LEI console or to a user-specified file name) and whether or not to include a timestamp with each trace log entry.

If there is a problem with your Activity or Connection, the Trace MetaConnector is a tool for you to use when troubleshooting. Also, trace output is useful for Lotus support staff when troubleshooting customer calls.

After launching an Activity that uses a Trace MetaConnection, the first piece of information seen is a “BEGIN TRACE” string along with a timestamp. The next sequence of calls is typically LCXIdentify, LCXSetProperty, and then LCXConnect. At this point you may see a number of different Activity-specific operations such as LCXExecute, LCXSelect, LCXInsert, LCXFetch, LCXUpdate, or LCXRemove. At the conclusion of an Activity you’ll usually see LCXDisconnect, followed by LCXTerminate, and then an “END TRACE” label.

### System Log File

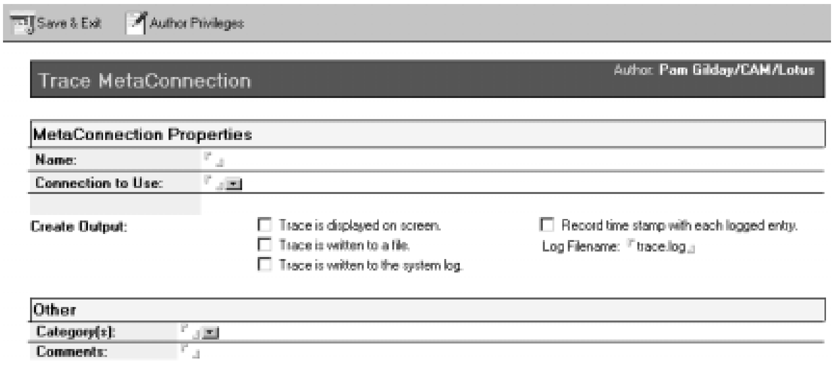
With the exception of beginning and ending trace timestamps, all trace information can be sent to the system log. Since the system specifies a timestamp for each entry, the loss of the beginning and ending trace timestamps are not meaningful.

Trace output using the stdout and log file methods uses carriage returns to format some information types, such as field list data. This formatting is not present when using the system log method for capturing trace output.

**Note** The system log seems to perform with a noticeable amount of delay. This is caused by the disk input-output operations that are required for making each log entry.

## Trace MetaConnection Document

The Trace MetaConnection document is shown below. Choose Create – MetaConnection – Trace to access the Trace MetaConnection document. The fields and options in the document are described in the table below.



Trace MetaConnection
Author: Pam Gilday/CAM/Lotus

MetaConnection Properties
Name:
Connection to Use:
Create Output:
☐ Trace is displayed on screen.
☐ Trace is written to a file.
☐ Trace is written to the system log.
☐ Record time stamp with each logged entry.
Log Filename: trace.log

Other
Category(s):
Comments:

| *Field* | *Description* |
|---|---|
| Name | Names the MetaConnection. |
| Connection to Use | Names the underlying Connection to be traced. |
| Trace is displayed on screen | Specifies whether or not to write trace data to the LEI console window. |
| Trace is written to a file | Allows you to capture trace data to a text file. All output captured to a log file is appended to the end of the file. If the output file does not exist, it will be created. If the log file cannot be opened, a message indicating this will be sent to the System (Activity) log.<br><br>The log file can grow quite large. You can delete the log file when the trace data is no longer needed.<br><br>The default log file name is trace.log. If you specify some other name (using the Log Filename option), such as "c:/foo/mytrace.log", then the initial output, up to the point where the file name parameter is read, will be sent to "trace.log", and trace output after that point will be logged to the "c:/foo/mytrace.log" file. |

*continued*

| *Field* | *Description* |
|---|---|
| Log Filename | User-specified log file name – allows you to specify a log file name. This option can provide more control over the trace output destination.<br><br>When a trace is started, initialization occurs followed by a series of LCXSetProperty calls based on information from the Connection document. It is only after this point that trace data can be logged to a user-specified file: the initial trace output is lost. |
| Trace is written to the system log | Determines whether or not output will be written into the Activity log. |
| Record time stamp with each logged entry | Records a timestamp with each entry that is captured in the log file. |

# Chapter 17
# Introduction to LEI Activities

This chapter provides an introduction to LEI Activity documents. It includes an introduction to Activity documents, descriptions of fields common to all Activities, Activity scheduling options, and information about command line execution of Activities.

## Introduction to Activity Documents

Activity documents contain the information that instructs the LEI server to execute an event, such as a data transfer. Each Activity type has a specific Notes form that can be used to create the Activity. Each LEI Activity is described briefly below. For more information, refer to the specific chapter for each Activity.

| *Activity* | *Description* |
|---|---|
| Admin-Backup | Backs up Administrator database data. |
| Admin-Purge Log | Purges the LEI logs. |
| Archive | Archives a database. |
| Command | Executes an action against a database. |
| Direct Transfer | Transfers data from one database to another. |
| Polling | Polls a database to see if a specified condition exists, and, if so, executes an Activity. |
| RealTime Notes | Catches and handles Notes events as they occur. |
| Replication | Synchronizes data in different databases. |
| Scripted | Executes an Activity whose data sources and function are defined by a LotusScript script. |
| Java | Executes a user-specified Java application. |

## Creating Activities

While building an Activity, you can choose connection information from existing Connection Documents. You should therefore create any Connection Document that you intend to use before you create the Activity. You should also test your Connections after defining them to ensure that they are properly defined. See Chapter 5, "Introduction to Connectors," for information about how to test your Connection.

Although each of the Activity documents contains information specific to the type of Activity, the Activity name, status, Activity options, and scheduling sections are common to all of them. This common information is documented in the following sections.

Activity documents use Notes collapsible sections to contain Activity option sets.

## Common Activity Fields and Buttons

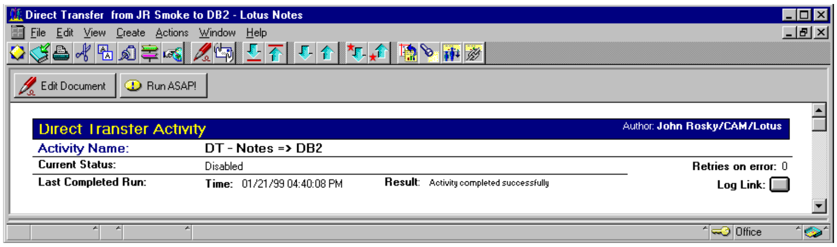
The fields and buttons described below are common to most Activity documents.

### Author Privileges Action Button

This button appears in the Action Button Toolbar for all Activities. It allows the author of the document to view and change author privileges.

### Activity Name and Status

The following status information appears at the top of every Activity document. It is read only, and cannot be modified directly.


Direct Transfer from JR Smoke to DB2 - Lotus Notes
File Edit View Create Actions Window Help
Edit Document
Run ASAP!
Office

Direct Transfer Activity    Author: John Rosky/CAM/Lotus

Activity Name: DT - Notes => DB2

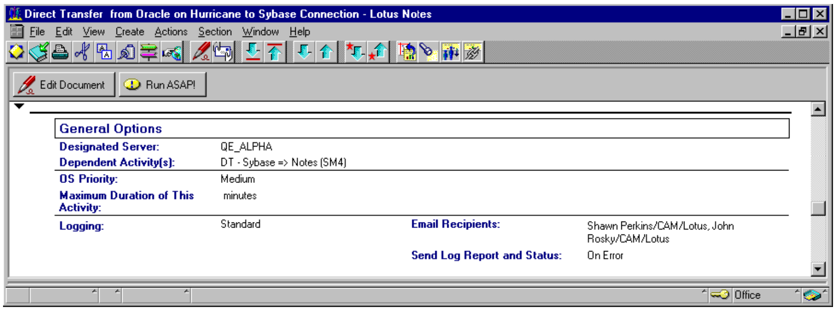
Current Status: Disabled    Retries on error: 0

Last Completed Run: Time: 01/21/99 04:40:08 PM    Result: Activity completed successfully    Log Link:

| *Field* | *Description* |
|---|---|
| Type of Activity | For example, Direct Transfer Activity. |
| Author | Person who created the Activity. |
| Activity Name | A user defined name and serves as a unique identifier for the Activity. This uniqueness is enforced through a validation process at the time the Activity is saved. |
| Current Status | The status of the document. This information is reported by the server itself and cannot be modified directly. The status can be any of the following:<br>New — The Activity is not saved. This status appears while the Activity is being built.<br>Not Yet Scheduled — The Activity's schedule has not been enabled since its creation.<br>Scheduled for…. — The Activity is enabled and scheduled to be executed at the time indicated.<br>Active on… — The Activity is currently running on the listed server.<br>Scheduling Disabled but Running ASAP — The Activity has been queued to run via the Run ASAP button but has not started running yet.<br>Disabled but Running ASAP — The Activity is disabled but is currently running as a result of being executed by clicking the Run ASAP button.<br>Disabled — The Activity's schedule is currently disabled. To enable the schedule, open the Activity document in Edit mode and change the Enabled/Disabled field to Schedule Enabled. |
| Last Computed Run | Show the time and result of the last run. |
| Time | Date and time that the Activity last ran (successfully or unsuccessfully). A zero means that the Activity has never been executed. |
| Result | What happened in the last run. If the Activity has never run, you'll see the caption "Activity has not yet run to completion." |
| Log Link | Click the Log document icon (if available) to see the log document for the last execution of this Activity. |

## General Options

The General Options fields let you define general Activity information. These options are described below.


Direct Transfer from Oracle on Hurricane to Sybase Connection - Lotus Notes
File Edit View Create Actions Section Window Help
Edit Document
Run ASAP!
Office

General Options

| | |
|---|---|
| Designated Server: | QE_ALPHA |
| Dependent Activity(s): | DT - Sybase => Notes (SM4) |
| OS Priority: | Medium |
| Maximum Duration of This Activity: | minutes |
| Logging: | Standard |
| Email Recipients: | Shawn Perkins/CAM/Lotus, John Rosky/CAM/Lotus |
| Send Log Report and Status: | On Error |

| *Field* | *Description* |
|---|---|
| Designated Server | If no LEI server is listed here, the first server to poll the control store when the Activity is scheduled to run gets to execute it. If a server is listed, then only that server can run the Activity. |
| Dependent Activity(s) | Other Activity or Activities that will run when the current Activity executes successfully. With dependent Activity(s) you can connect the execution of two or more Activities. Activities must be defined (in their own documents) before being added as a dependent.<br><br>Place the cursor in the edit field and press Enter to see a list of available Activities to choose from. |
| OS Priority | Low, Medium, or High priority. In a multi-tasking environment the LEI Server machine allocates resources based on priorities. High priority would gain more LEI resources than Medium or Low. |
| Maximum Duration of Activity | Sets a time-limit in minutes on how long an Activity can run before being terminated. A setting defined here overrides the maximum duration time set in the server configuration for the server that executes the Activity. Zero stands for no timeout. |
| Logging | Standard performs normal logging, creating an Activity log document for each execution of this Activity. Select None to avoid logging executions of this Activity.<br><br>Buffered Logging – If Buffered Logging is selected, log entries are stored in memory until the conclusion of the Activity. When the Activity is complete, log entries are written to the log database.<br><br>Unbuffered Logging – Log entries are written to the log database as they are generated. |

*continued*

| *Field* | *Description* |
|---|---|
| Activity Retry on Error | Use this option to specify how many times to retry running the Activity at intervals specified in minutes. This option is useful in cases where the designated server may not be available to service the first request to run the Activity, so you would use this option to automatically keep retrying. |
| Mail Recipients | Enter the Notes user names of the people who should receive the e-mail notification. Commas should separate the names. |
| Send Log Report and Status | Whether mail should be sent Never, Always, or On Error. With the On Error option, mail is sent only when the Activity does not complete successfully. |

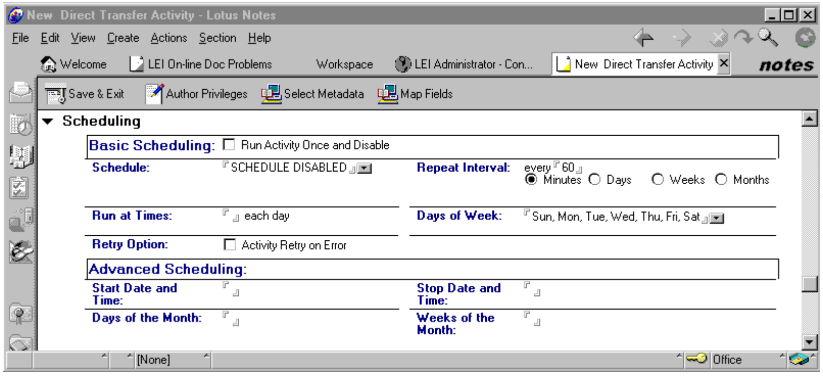
### Category and Comments

The Category and Comments fields let you categorize documents and add Comments.

| *Field* | *Description* |
|---|---|
| Category | Entering a category name will let you group a set of documents by category in the view Activities by Category. If the category you enter doesn't exist, it will automatically be created. You can only enter a single category per document. |
| Comments | Whatever you want to put here to describe the document. This is a rich text field, so that you can place anything here including document links. |

## Activity Scheduling Options

The Scheduling section of an Activity, shown below, defines the schedule according to which the Activity is to be executed.


New Direct Transfer Activity - Lotus Notes
File Edit View Create Actions Section Help
Welcome
LEI On-line Doc Problems
Workspace
LEI Administrator - Con...
New Direct Transfer Activity
notes
Save & Exit
Author Privileges
Select Metadata
Map Fields
[None]
Office

Scheduling

Basic Scheduling: ☐ Run Activity Once and Disable

Schedule: SCHEDULE DISABLED

Repeat Interval: every 60 [✓ Minutes] Days Weeks Months

Run at Times: each day

Days of Week: Sun, Mon, Tue, Wed, Thu, Fri, Sat

Retry Option: ☐ Activity Retry on Error

Advanced Scheduling:

Start Date and Time:

Stop Date and Time:

Days of the Month:

Weeks of the Month:

## Overview of Scheduling

The scheduling section of Activity documents is common for all Activities. The parameters affect the scheduling in a specific order. Additionally, the Start Activity icon in the Navigator or the Run ASAP action button each mark the Activity to run as soon as possible regardless of the schedule. Neither of these affect the scheduled run time. It is important to note that the scheduling parameters affect when an Activity is scheduled to start. Additionally, for the Polling and RealTime Activities, the scheduling information is used to determine when it should stop. More information about how the scheduling settings affect the stop time may be found at the end of this section.

The first parameter of scheduling is the DISABLED/ENABLED/RESTRICT setting. When an Activity is disabled, all of the other scheduling settings are ignored. When enabled, the Activity will be scheduled to run according to the setting of the remaining parameters. The Activity may actually run some time after its schedule time if the LEI server is not available either because it is shut down or it is busy. In this case, the Activity will run as soon as possible after the server becomes available. If the Activity should not run except at the schedule times, the RESTRICT TO SCHEDULE should be used.

The next parameters to affect the scheduled run time are the Start Date and Time and Stop Date and Time. These create boundaries for the run time. Scheduling will not begin before the start date and will not continue after the stop date. As an example, a start date of June 6, 1999 and a stop date of December 25, 1999 would prevent the Activity from being scheduled any time before 00:00:00 AM on the 6th of June and would prevent it from being scheduled any time after 12:59:59 PM on the 25th of December, 1999. Note: If only a time value is indicated for these parameters, then the current date is assumed.

Once the boundaries of scheduling are established, the Days of Week, Weeks of the Month, and Days of the Month settings are used together to determine which dates in the month are acceptable. First, any dates which do not fall on the selected Days of the Week, Days of the Month, or Weeks of the Month are ignored. For the purposes of scheduling, a week is a seven day period starting with the first of the month. As an example, if the first of the month is a Thursday, then each week is Thursday through Wednesday. Both the Weeks of the Month and Days of the Month choices will accept negative numbers to indicate counting from the end of the month. This feature is helpful for scheduling Activities to run on the last day of the month or even the last Friday of the month. For example, to schedule on the last day of every month, set Days of the Month to -1; to schedule on the last Friday of the month, set Weeks of the Month to -1, Days of the Week to Friday.

The final parameters for scheduling are Repeat Interval and Run at Times. When specified, any times which fall outside the Run at Times settings are ignored. The values may be any combination of distinct times such as 8:00 AM, 11:30 PM and time ranges such as 7:00 AM - 3:00 PM. Once the initial run time has been computed, subsequent run times are scheduled at the repeat interval. To maintain accuracy of the scheduling interval, an Activities 'next' run time is computed at the start of the current execution. This prevents variation in the amount of time it takes an Activity to complete from effecting its next scheduled run time. If an Activity does not complete until after the next scheduled run time, it will complete the current execution and then restart immediately to complete the overdue scheduled run.

For example, an Activity has a repeat interval of 60 minutes and typically takes only few minutes to complete. The Activity will run on its hourly interval such as 6:30 PM, 7:30 PM, etc. However, if the second execution takes more than 60 minutes to complete, the Activity will complete the 7:30 PM run after 8:30 PM. It will then start the 8:30 PM run as soon as the previous run is complete and, assuming it returned to its typical execution of only a few minutes, would be scheduled for 9:30 PM. When an Activity repeatedly overruns its next start time, the scheduling is adjusted to prevent continually scheduling overdue times.

All of the scheduling settings are used to determine when the Activity will start. For many Activities, they start according to the schedule and then run to completion. The Polling and RealTime Activities also use the scheduling settings to determine when to stop. The Run at Times, Days of the Week, Weeks of the Month, and Days of the Month settings are used to determine valid start times. These same values are used to determine invalid times. For Activities which require a stop time, the first invalid time following the start time is computed and used as the stop time.

**Note** If no values are specified for Run at Times and Days of the Month and Weeks of the Month, and every weekday is listed in Days of the Week, then there are no invalid times and the Activity will not have a stop time. Also, the start time is computed before the stop time. As described previously, the start time is computed using the Repeat Interval. It is possible to have the next start time occur before the current stop time. For example, a Notes RealTime Activity with valid run times being 7:00 AM - 3:00 PM and a repeat interval of 60 minutes would schedule to start at 7:00 AM and stop at 3:00 PM. However, it would have its next run time computed as 8:00 AM. When the Activity completed at 3:00 PM, it would view the 8:00 AM start time as overdue, and restart immediately. You can use the Restrict to Schedule setting to prevent this behavior. An alternative is to set the repeat interval to 1 day, which causes the run time to be 7:00

AM each day. Using Restrict to Schedule and setting the repeat interval to a small value has the benefit of restarting the Activity immediately in the event it stopped prior to the computed stop time.

## Scheduling Process

The scheduling process allows you to set two kinds of controls:

- The window of time during which the Activity can take place. This window can amount to an absolute start and stop date and time or to a regularly defined period (such as every week day between 7:00 AM and 6:00 PM).
- The Activity interval. The interval defines how often the Activity will be executed within the schedule window, which can be from every minute to every few months.

An Activity executes as soon as the window opens and on schedule thereafter. For example, an Activity scheduled to execute once an hour between 7:30 AM and 5:00 PM will first execute at 7:30, then at 8:30, and every hour until its final execution at 4:30 PM.

To avoid scheduling inconsistencies between servers, the LEI administrator supplies a standard time to the LEI server when it is started. As a result, all LEI servers supported by a single administrator operate on the same time. LEI takes Daylight Savings Time into consideration when necessary.

**Note** If you want the Activity to take place when an event occurs rather than on schedule, make it the dependent Activity of a Polling Activity. See Chapter 24 “Polling Activity,” for more information.

### Basic Scheduling

The following fields let you define when and how often an Activity is executed:

**Run Activity Once and Disable —** Causes the Activity to run only once on schedule when checked. If scheduled, the Activity will run on schedule but then will not run again. After it is executed its Enabled/Disabled condition is changed to Schedule Disabled. Using Run ASAP to execute an Activity does not count towards its Run Once status. Run Once is useful for testing new Activities.

**Schedule** — There are three options for LEI scheduling: Schedule Disabled, Schedule Enabled and Restrict to Schedule. With the cursor in the edit field, press the space bar to toggle between the three values.

- **Schedule Disabled**— Will not run based on any of the remaining scheduling information, but can still be run by clicking the Run ASAP button. Even though the Activity is currently "running ASAP," the schedule status will still be shown as disabled. Shared execution also remains available.
- **Schedule Enabled** — Will run based on the remaining scheduling information.
- **Restrict to Schedule** — Similar to Schedule Enabled, but if the Activity execution starts outside a valid execution cycle, the Activity will not run and will be rescheduled for the next valid execution time. This option is used when an Activity should not execute with a delayed start, such as when the scheduled run time for an Activity passed while the LEI Server was down. When the server is started, overdue Enabled Activities start, but Activities marked Restrict to Schedule do not run until the next scheduled time.

  **Note** Activities are automatically disabled after a scheduled run if the Run Once option is checked.

**Repeat Interval** – The frequency of execution. The interval can be any number of minutes, days, weeks, or months. (Hours are entered as multiples of minutes.)

Once an Activity is executed, its next run time is calculated by adding the interval to the last starting run time, as long as it falls within the window of execution defined in the Days of Week and Run at Times fields.

If an interval of once a day or greater is used and no time of day window is defined, the next execution will be scheduled for midnight at the start of the next valid day.

**Run at Times** – The window of time during which the Activity will be executed. The delimiting times can be entered in AM/PM or 24-hour format and separated by a hyphen: 5:00 AM - 6:00 PM or 5:00 - 18:00 or separated by commas (the combination 4:00, 8:00 - 10:00 is valid). In either case, you must use the format hours:minutes (5 PM and 1700 are not valid).

If no times are entered, then the Activity can take place at all times.

**Days of Week –** The days of the week on which the Activity will be executed. The Activity will be run during the Times of Day only on the days listed.

To choose days, place the cursor in the day field and press Enter. Choose from the list. You can also add or delete directly: Press the first letter of the day to insert it, making sure that days are separated by commas.

**Retry Options –** Enables you to specify a retry interval should the Activity fail prior to or during execution.

### Advanced Scheduling

The Advanced Scheduling options enable you to schedule an Activity to execute on certain days or weeks of the month. It also allows you to specify a time in conjunction with a date.

When both Days of the Month and Weeks of the Month are defined, both sets are valid. In other words, if weeks 1 and 2 and days 26, 27, and 28 are defined, then the Activity will be available for execution on days 1 through 14 plus 26, 27, and 28. The Activities, however, will always be limited to the Days of Week listed in the basic scheduling section. If the only listed day of the week is Monday, then the Activity will take place only when the days defined in the Advanced Scheduling section fall on a Monday.

**Start/Stop Date and Time –** The dates and times that define the span of time during which the Activity may be scheduled to execute.

You can set one or the other or both. A date and no time defaults to midnight. When just a time with no date is entered, the current date is used. If you intend to enter a start or stop date/time, however, it's recommended that you enter both since the Administrator will not supply the missing information into the entry field.

Enter a start or stop date/time only if you want to set a limit on the span of time that the Activity should be scheduled to start running. You can leave a date/time blank to set no start or stop limit on the Activity schedule. If you leave the start field blank, the Activity is first run when the document is completed and saved.

An Activity that is running at Stop Time will continue to run until it completes, except for a Polling Activity, which, once started, runs up to Stop Time. If there is no stop time for the Polling Activity, it can still be stopped by using the Close command on the LEI Server console.

The date and time format to be used is specific to the international settings and the operating system on which the Notes Client is running.

**Days of the Month** – This option specifies on which days of the month the Activity can be executed. Day 1 is the first day of the month. The days should be listed separated by commas. Days entered as negative numbers count back from the last day of the month with -1 being the last day. If no days are listed, all days are valid (within the scope of other time-span definitions).

For example: 7, 15, 21, -1 means that the Activity can be executed only on the seventh, fifteenth, twenty-first, and last day of the month.

Range formats, such as 10 – 20, cannot be used. Each day in a range must be listed.

**Weeks of the Month** – This option specifies on which weeks of the month the Activity can be executed. The weeks should be listed separated by commas. Weeks entered as negative numbers count back with -1 being the last week (the last seven days of the month). If no weeks are listed, all weeks are valid (within the scope of other time-span definitions).

Weeks are seven day intervals starting with the first day of the month, which may not necessarily be a Sunday to Saturday interval, and are defined as follows:

- Week 1 – Days 1, 2, 3, 4, 5, 6, 7
- Week 2 – Days 8, 9, 10, 11, 12, 13, 14
- Week 3 – Days 15, 16, 17, 18, 19, 20, 21
- Week 4 – Days 22, 23, 24, 25, 26, 27, 28
- Week 5 – Days (29), (30), (31)

## Command Line Execution of Activities

LEI allows Activities to be executed from the system command line of the machine where an LEI server is running on a host. You might use this capability to execute LEIACT in response to specific database events.

Command syntax for starting an LEI Activity from the UNIX command line is as follows:

```
leiact "Activity Name"
```

Command syntax for starting an LEI Activity from the Windows command line is as follows:

```
nleiact "Activity Name"
```

In both cases, “Activity Name” is the name of the existing LEI Activity that you want to run.

For example, the following NT command line would invoke the quoted Activity name:

```
nleiact "Michaels Direct Transfer Activity"
```

When an Activity is started from the system command line, it does not appear as running in the LEI Administrator or on the server console. However, logging still occurs.

The LEI server only runs Activities that were scheduled from the Administrator (including those marked Run ASAP). Also, when an Activity is running from an LEI server, it will not run simultaneously on that or any other server until it has completed its current execution. This is referred to as *exclusive* execution. In contrast, when an Activity is started from the system command line, its execution may be “shared.” This means that the Activity may be run more than once simultaneously. At first this may not seem very safe or useful but there are special cases where this is important; performing searches for an Internet Web application or generating reports are two examples where the *shared* execution is valuable.

When running an Activity from the command line (with leiact.exe or leicgi.exe), please note the following:

- Any dependent Activities listed in the Activity document are not executed.
- The Activity document’s document link to the log database is not updated to point to the new log entry for that run.

**Note** The LEICGI functionality is described in Appendix B, “Common Gateway Interface (CGI) for LEI.”

## Command Line Execution of Activities on AS/400

An LEI Activity can be launched from the AS/400 command line.

If you are authorized to run the program QNOTESLEI/LEIACT, you can launch an LEI Activity from an AS/400 command line and it will run under that process. Input (in single quoted string) would be the name of the LEI Activity to be executed. Note that the LEI Activity will access DB2/400 data under the AS/400 user profile registered in the DB2 Connection and will access Notes data under the user identified in KeyFileName. If the user ID in KeyFileName is password protected, you will be prompted for the password within your interactive AS/400 process.

Note that LEIACT in LEI is public *EXCLUDE and should remain so as to secure Activities from being launched by unauthorized users.

Our experience is that performance of an Activity will be better when launched under the umbrella of the LEI Server. Also note that you will get a log entry for all Activities from a call to LEIACT, but the status of that execution will not be recorded in the LEI Activity itself (under last run status).

**Example:**

```
CALL  QNOTESLEI/LEIACT    'Transfer Open Issues'
```

## Pop-up Administrator Help

Every LEI Activity document includes pop-up help for each of the sections and most of the fields. Dark blue text indicates that pop-up help is available; click the dark blue text and hold the mouse button down to view the pop-up help for a section.

## Data Mapping

During a Direct Transfer or Replication Activity, LEI maps columns in the source and target databases in one of three ways – by position, by field name or by user-defined mapping. Understanding how LEI performs field mapping is particularly useful when configuring your Replication and Direct Transfer Activities.

**Note** For information about data types, see Appendix A, "Lotus Enterprise Integrator and Data".

### According to Field Names

Data is transferred from a field in the source database to a field of the same name in the target. If the Activity's command statement is a SQL query, you can use column aliases to match the source field to the destination: for example "SELECT empno employee" maps the empno field in the source to the employee field in the target.

**Note** The actual expression depends on the database SQL syntax.

### According to Field Position

Data in the first field of the source result set is copied into the first field of the target database, and so on. If the command statement in the Activity selects a subset of columns from the metadata, they will be treated as a simple incremental series starting with 1. The order in which the columns are listed in the selection statement is the order in which they will be transferred. The SQL statement "SELECT column6, column4, column7 FROM table" maps column 6 of the source to field 1 of the target, column 4 to field 2, and column 7 to field 3.

### According to User Definition

You supply field names for both the source and target, and data fields are mapped in the order provided. For example, source fields listed as "A, B, C" and target fields listed as "X, Y, Z" result in source field A moving to target field X, B to Y, and C to Z.

### When Field Numbers Don't Match

Since Replication does not require the same number of columns in both databases, mismatching numbers of columns only affects Direct Transfer Activities. The following occurs with these Activities:

- More columns in the source result set than in the target metadata – The transfer will fail and you'll get an error message.
- Fewer columns in the source result set than in the target metadata – The transfer will succeed and the "orphan" columns in the target will be filled with null values.

## Moving Text List Data over Different Platforms

In Activities such as RealTime, Notes databases store all data in the same canonical format (little-endian byte ordering) on all platforms. Notes Data is automatically converted by LEI to the host platform's byte ordering format when read (for example, when the key field in the Notes form is accessed and used for record queries on the host back end system), and back to Notes canonical format when saved to a Notes form. (Composite data is an exception: It is always left in canonical format).

**System Planning Considerations**: When moving data between Domino servers and between Notes and other database systems, there is normally not a problem. In the case, however, where data is moved from one Domino server to another through another DBMS using different LEI server platforms, multi-value types (text list, number list, datetime list) become corrupted. For example, you use an NT LEI server to move a text list from Notes to DB2. If the data is moved from DB2 back to Notes through a second LEI server running under Windows NT, there is no problem. But if the second LEI server runs on a UNIX platform, the text list data in the destination database will be invalid.

# Chapter 18
# Admin-Backup Activity

This chapter provides information about the LEI Admin-Backup Activity. Included is information about what this Activity is, when to use it, and how to create the Activity.

## Introduction to the Admin-Backup Activity

The Admin-Backup Activity creates a backup copy of your LEI Administrator database (leiadm.nsf), and, optionally, the Script vault database (leivlt.nsf).

## When to Use the Admin-Backup Activity

Use the Admin-Backup Activity when you want to create a safe backup of the LEI Administrator and associated documents.

You should use the Admin-Backup Activity at regular intervals to ensure that you always have backup copies of LEI Activities, Connections, and Configurations that have been created. You can also use the Admin-Backup Activity when your Administrator database becomes cluttered with too many documents and you want to clean up the database while at the same time keeping copies of everything already there.

## How to Create an Admin-Backup Activity

The steps below outline the general procedure for creating an Admin-Backup Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

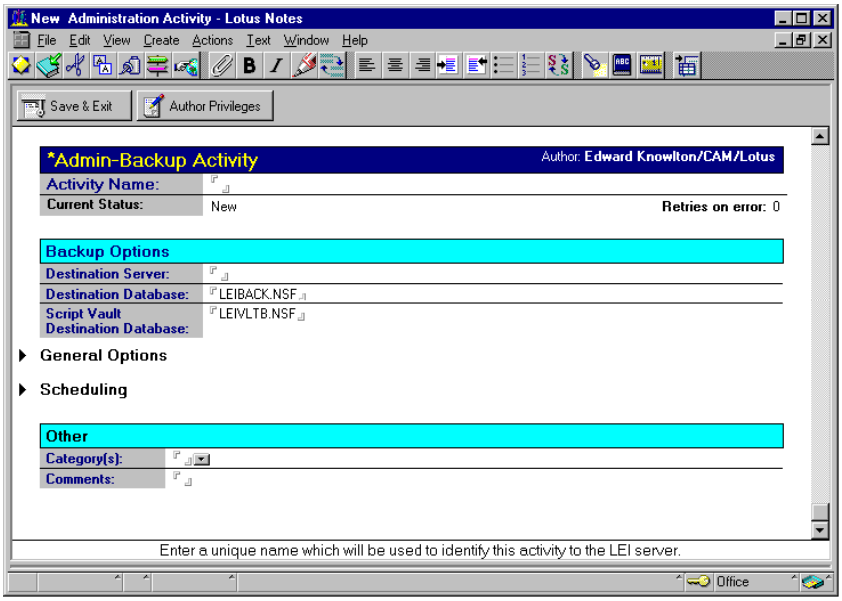
1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – Other and then select Activity/*Admin-Backup from the list in the dialog box that appears, or click Create Activity on the Navigator and select Admin-Backup.

3. Fill in the required fields and select the desired options for the Activity in the Admin-Backup Activity document, including its scheduling options.
4. Save the Activity document.

You can select the Run ASAP option to have the Activity executed by the LEI server as soon as possible after saving the Activity.

## The Admin-Backup Activity Document

Choose Create – Activity – Other and then select Activity/*Admin-Backup from the list in the dialog box, or click Create Activity on the Navigator and select Admin-Backup. The Admin-Backup Activity document shown below appears. Use this document to define the Admin-Backup Activity. The fields in this document are described below.


New Administration Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges

*Admin-Backup Activity Author: Edward Knowlton/CAM/Lotus

| | |
|---|---|
| Activity Name: | |
| Current Status: | New Retries on error: 0 |

Backup Options

| | |
|---|---|
| Destination Server: | |
| Destination Database: | LEIBACK.NSF |
| Script Vault Destination Database: | LEIVLTB.NSF |

General Options

Scheduling

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

Enter a unique name which will be used to identify this activity to the LEI server.

See Chapter 17, “Introduction to LEI Activities,” for a description of the following sections that are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

## Backup Options

| *Field* | *Description* |
|---|---|
| Destination Server | The name of the Domino server where the backup copy of the Administrator database will be stored.<br>You must specify a target server in order for this Activity to run properly. |
| Destination Database | The file name of the backup copy (a Notes .nsf file). If the database does not exist, LEI will create it. Otherwise, the existing database will be overwritten. In addition, the file path can be assigned, for example, \qe\leiback.nsf.<br>The path name must be specified as relative to the data directory. |
| Script Vault Destination Database | The file name of the Script Vault copy (a Notes .nsf file). If the database does not exist, LEI will create it. Otherwise, the existing database will be overwritten. In addition, the file path can be assigned, for example, \qe\leivltb.nsf.<br>The path name must be specified as relative to the data directory.<br>Leave this field blank if you do not want to back up the Script Vault. |

**Note** The LEI server must have Manager access to the backup database in order to delete the previous backup. Since the backup's ACL is copied from the Administrator database's ACL, the server essentially needs Manager access to the LEI Administrator.

# Chapter 19
# Admin-Purge Log Activity

This chapter provides information about the LEI Admin-Purge Log Activity. Included is information about what this Activity is, when and why to use it, and how to create the Activity.

## Introduction to the Admin-Purge Log Activity

Use the LEI Admin-Purge Log Activity to purge the LEI log database of documents older than a user-specified number of days.

## When to Use the Admin-Purge Log Activity

You should use the Admin-Purge Log Activity for the LEI logs at different times, depending on your situation. For example, if the LEI logs take up too much disk space in your environment, you may want to run this Activity on a fairly regular schedule.

## How to Create an Admin-Purge Log Activity

The steps below outline the general procedure for creating an Admin-Purge Log Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – Other and then select Activity/*Admin-Purge Log from the list in the dialog box that appears, or click the Create Activity icon on the Navigator and select Admin-Purge.
3. Fill in the required fields and select the desired options for the Activity in the Admin-Purge Log Activity document, including its Scheduling options.
4. Save the Activity document.

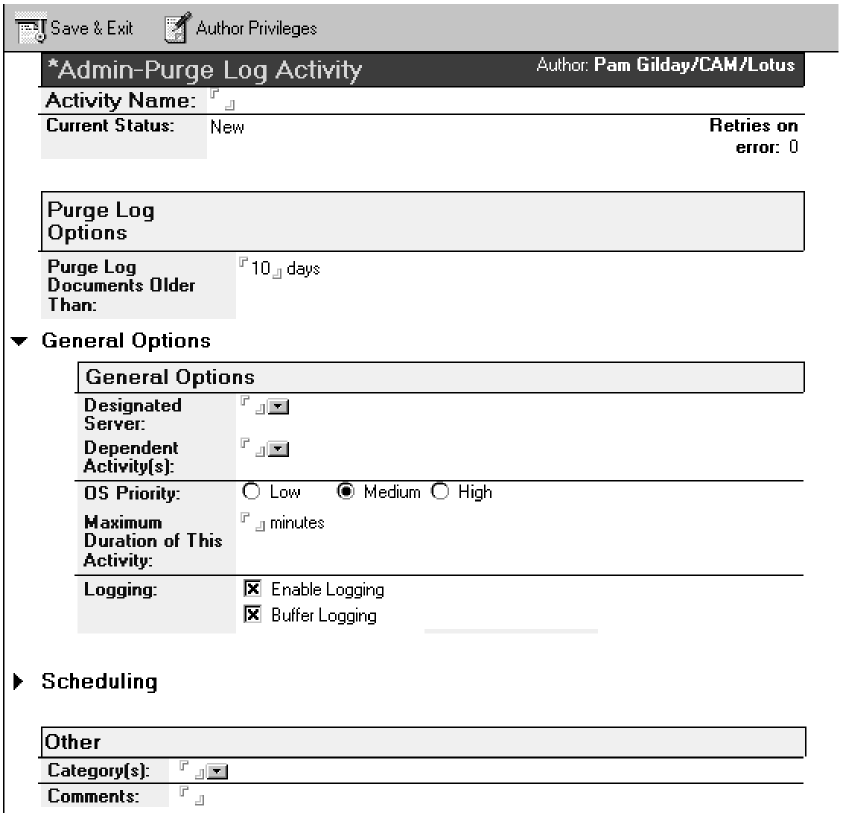
You can select the Run ASAP option to have the Activity executed by the LEI Server as soon as possible after saving the Activity.

## Saving Selected Logs before Purging

In order to save any logs, you must copy them to another Notes database before running the Admin-Purge Log Activity.

## The Admin-Purge Log Activity Document

Choose Create – Activity – Other and then select Activity/*Admin-Purge Log from the list in the dialog box that appears, or click Create Activity on the navigator and select Admin-Purge. The Admin-Purge Log Activity document shown below appears. Use this document to define the Admin-Purge Log Activity.



*Admin-Purge Log Activity — Author: Pam Gilday/CAM/Lotus

Activity Name:

Current Status: New — Retries on error: 0

Purge Log Options

Purge Log Documents Older Than: 10 days

General Options

Designated Server:

Dependent Activity(s):

OS Priority: Low [✓ Medium] High

Maximum Duration of This Activity: minutes

Logging: [✓ Enable Logging] [✓ Buffer Logging]

Scheduling

Other

Category(s):

Comments:

See Chapter 17, “Introduction to LEI Activities,” for a description of fields that are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

## Admin-Purge Log Options

| *Field* | *Description* |
|---|---|
| Purge log documents older than | The cut off age for log records. All documents older than the specified number of days will be deleted from the log.<br><br>**Note** The number of days are counted from the time and date that the Activity is run. Each day is considered a 24 hour period. |

# Chapter 20
# Archive Activity

This chapter provides information about the LEI Archive Activity. Included is information about the Activity, when to use it, and how to create Archive Activities.

## Introduction to the Archive Activity

An Archive Activity moves data from one database to another. As records are moved into the target database, they are deleted from the source; one record at a time.

The source data and target location are indicated by metadata name (for example, table, form, etc.). Selection of documents to archive can be done with either a condition or a relative time stamp.

The Archive Activity deletes the original records from the source database.

## When to Use the Archive Activity

You can use the Archive Activity to protect data on a regular basis, to put in long term storage data that is infrequently accessed, or to archive data to a removable storage device to free space on a system server.

Some reasons to use the Archive Activity include:

- To archive infrequently accessed data.
- When migrating from one database to another.
- To create space on a full disk.

### Combining an Archive Activity with other Activities

You can use the Archive Activity in combination with other LEI Activities. For example, you can combine the Archive Activity with a Polling Activity. You could then poll your system databases to determine when data was last accessed, and then archive it based on a specified date of its last access.

## How to Create an Archive Activity

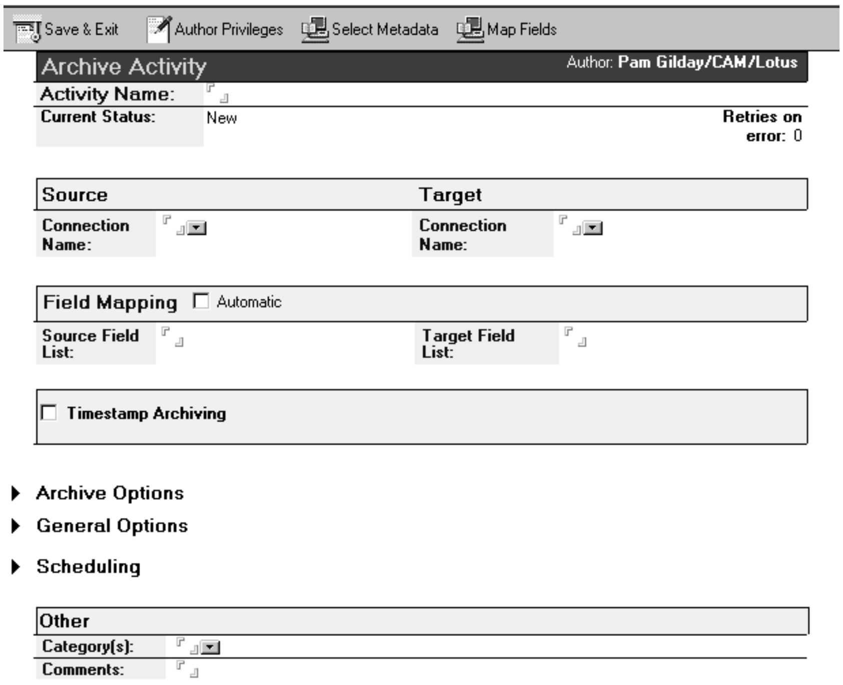
The steps below outline the general procedure for creating an Archive Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – Archive from the menu bar, or click Create Activity (in the Navigator) and select the Archive Activity type.
3. Fill in the required fields and select the desired options for the Activity in the Archive Activity document, including its Connectors and Scheduling options. Note that any Connectors you want to use in the Activity must already exist and should already have been tested using the LCTEST described in Chapter 5, "Introduction to Connectors."
4. Save the Activity document.

You can select the Run ASAP option to have the Activity executed by the LEI server as soon as possible after saving the Activity.

## The Archive Activity Document

Choose Create – Activity – Archive from the menu bar, or click Create Activity on the Navigator to access the Archive Activity document, shown below. The Archive document defines the information needed for the Archive Activity. The fields and options in the Archive document are described in the following sections.


Save & Exit Author Privileges Select Metadata Map Fields

Archive Activity Author: Pam Gilday/CAM/Lotus

Activity Name:

Current Status: New

Retries on error: 0

| Source | | Target | |
|---|---|---|---|
| Connection Name: | | Connection Name: | |

Field Mapping ☐ Automatic

| | | | |
|---|---|---|---|
| Source Field List: | | Target Field List: | |

☐ Timestamp Archiving

▸ Archive Options

▸ General Options

▸ Scheduling

Other

Category(s):

Comments:

See Chapter 17, “Introduction to LEI Activities”, for a description of the following sections that are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

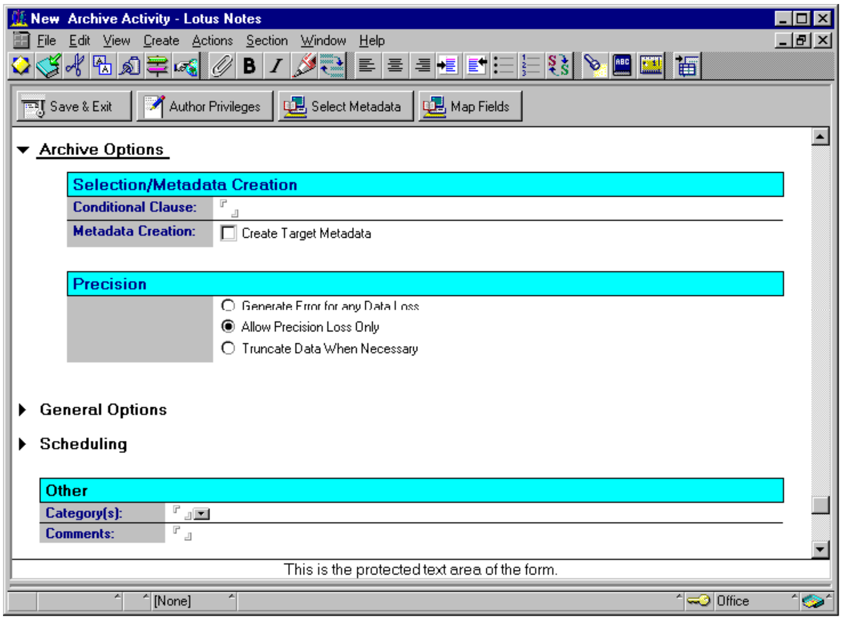
## Connection Properties

The Connections section of the Activity form provides information on the data source and data destination. Connections can exist between the LEI Server and data sources. These Connections must be created before they can be used in an Activity document.

| *Field* | *Description* |
|---|---|
| Edit Connection | The Edit Connection button (located in the title bar directly over the Connection's name) opens the selected Connection document in EDIT mode. This button is available only after you have entered a Connection name. |
| Connection Name | Names of the two Connections. The data is transferred from the Source Connection database to the Target Connection database. The Connection provides the server and database with access information.<br>To select a Connection, place the cursor in the Edit field and press Enter to see the list of Connections. You can also place the cursor in the Edit field and press a letter key to see the first Connection beginning with that letter. Repeatedly pressing a letter key scrolls through all Connections with that initial letter.<br>After entering the Connections, the Table Name field displays. |
| Table Name | After you enter a Connection name in the Connection Name field, the Table Name field displays. Enter the name of the table you want or click the Select Metadata button to select from a list of existing tables.<br>**Note** To create a table in the Target, you can enter a name in the Target Table Name field and in the Archive Options section (see below), you can click "Create Target Metadata." |
| Field Mapping<br>• Source Field List<br>• Target Field List | Enter the names of the fields in the Source Field List and in the Target Field List in the proper mapping order. The first source field maps to the first target field, the second to the second, the third to the third, and so on. This allows you to map any source field to any target field. |
| Timestamp Archiving | Allows you to pick a field with a time stamp so that you can use the date and time stamp as a criteria for archiving. You can use this in conjunction with the Archive Options to archive data. |

## Archive Options

Check any of the following options that you want to use, including one of the options pertaining to data loss.


New Archive Activity - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields

Archive Options

Selection/Metadata Creation

Conditional Clause:

Metadata Creation: [ ] Create Target Metadata

Precision

○ Generate Error for any Data Loss
[✓ ● Allow Precision Loss Only]
○ Truncate Data When Necessary

General Options

Scheduling

Other

Category(s):

Comments:

This is the protected text area of the form.

| *Field/Option* | *Description* |
|---|---|
| Conditional Clause | Allows you to use conditional criteria to choose specific record names in a table. |
| Metadata Creation | If you mark the "Create Target Metadata" box, LEI will create a new metadata in the database identified in the Target to hold the data being transferred if one does not already exist. In creating a new metadata, LEI will try to match the source metadata as much as possible.<br><br>**Note** If you do not mark the box, LEI will give you an error message if a specified target table does not exist. |
| Generate Error for any Data Loss | LEI writes an error to the log for any data that is lost as a result of the transfer and terminates the transfer. |
| Allow Precision Loss Only | LEI does not report loss of numerical or datetime precision as a result of the transfer. This option is the default. |
| Truncate Data When Necessary | LEI truncates text data when necessary to conform to field lengths in the destination database. Because of performance penalties, the truncate data option does not affect numbers being written. |

# Chapter 21
# Authority Propagation Activity

This chapter provides information about the Authority Propagation Activity. This Activity is specific to LEI running on the AS/400 platform.

## Introduction to the Authority Propagation Activity

An Authority Propagation Activity transfers database authorities from one database to another. Note that this is authority propagation, not two way replication. This Activity is unique to the AS/400. The authorities to be transferred are mapped between the two different database authority mechanisms (Domino and DB2/400) by the authority table below. This authority propagation can be executed ASAP or scheduled.

## Authority Propagation Activity Document

The Authority Propagation document contains information needed for the transfer of authorities:

- Connections indicating the source and destination databases
- Options controlling the transfer of the authorities
- Scheduling

Source and destination databases, including access information, must be defined in Connection documents before the Authority Propagation Activity is created. Also for Authority Propagation, both databases must be local to the LEI server. The Activity is only valid for Notes and DB2 Connections.

### Information in an Authority Propagation Activity Document

This section has fields common to most Activities, see Chapter 17, "Introduction to LEI Activities", for details.

### Propagate Activity

**Activity Name:** The name identifying this Authority Propagation Activity. You must enter an Activity name before entering any other information.

## Connections

Information on the data source and destination. These are Connections between the LEI Server and data sources and must be created before they can be used in an Activity document.

**Connection Names**: Names of the two Connections. The authorization is transferred from the Source Connection database to the Destination Connection database. The Connection identifies the server and database with access information. The Additional Information segment of the document (see below) identifies the metadata. For this Connection, metadata does not apply for Notes Connections. A DB2/400 table name is required for DB2 Connections.

To select a Connection, place the cursor in the edit field and press Enter to see the list of Connections. You can also place the cursor in the edit field and press a letter key to see the first Connection beginning with that letter. Repeatedly pressing a letter scrolls through all Connections with that initial letter.

**Default**: Once the Connections are selected, information about those Connections appears in the form. A default of N/A means that the information is that field is not used by the database type.

**Metadata**: The source and destination metadata. The metadata is only required for the DB2 Connection. For a Notes Connection, identification of the .nsf database is all that is required.

## Authorization Options

Check any of the following options:

**Full Replace of the Authority**: LEI will fully replace the current destination database's authority with the authorities materialized and mapped from the source database.

**Append to Existing Authority**: Checking this option causes LEI to update the authorities of the destination database to whatever authorities are materialized from the source database. But, if there are additional authorized users in the target that are not in the source database, they will remain as is. This means it does not clear the target authorities out before applying the new authorities.

### Log Conflict

Logging conflicts is the default and is recommended. As the application of authorities between two heterogeneous databases requires mapping of user information in one database domain (on the AS/400, this would be user profiles) to users in another database's domain (in Notes, this would be ACL entries), sometimes the mapping cannot be performed. LoggingConflicts will list those users by ACL entry or User profile whose authority could not be propagated.

### Necessary Authority

Note that the requesting user must have enough authority to materialize on the source side and apply authorities on the target side. In this case, this is *MANAGER for Notes databases and *ALL authority for DB2/400 databases. If the requester's authority is diminished as a result of running the authority propagation, a subsequent run of the request will fail.

## AS/400 Group and Authorization List Authorities

At the present time group authorities are only propagated one way, from DB2/400 to Notes. This is done by materializing the members of the group profile on the AS/400. Authority lists are also propagated from DB2/400 to Notes. Note that on the AS/400, Group authorities are accumulative but do not override explicit authorities.

## Mapping Notes ACL User IDs to AS/400 User Profiles

The AS/400 System Distribution Directory (SDD) and the Notes Name and Address Book (NAB) are used to map the names of users in one domain (AS/400 user or Domino user) to the other domain (Domino user or AS/400 user). When propagating authority from DB2/400 table to a Notes database, the authorized user profiles on the AS/400 are looked up in the System Distribution Directory and if found, the last name, first name (if specified) and middle initial (if specified) are used to generate an ACL entry that is then verified in the Name and Address Book. Any time there is no hit in the SDD, or the mapping is ambiguous, or the generated ACL is not found in the Name and Address Book, an informational message is logged in LEI. When propagating from a Notes database to DB2/400 table, the existing ACL entries for people only are parsed for last name, first name and optional middle initial, and these values are queried for in the SDD in order to associate a user profile with that ACL entry.

The Domino for AS/400 server install option 1 (AS/400 Integration) includes the Domino for AS/400 Directory Synchronization feature. This function can be used to provide automatic synchronization between the NAB and the SDD. This synchronization service by default is be set up to include Last, First and Middle name (where Middle may be optional depending on your location's user ID naming strategy). See the Domino for AS/400 documentation for more information on how to activate Directory Synchronization to populate and maintain the relationship between your SDD and NAB according to your naming policy. If you do not use Directory Synchronization, you can still use this Activity provided there is some correlation between the name fields in the SDD (for a user profile) and the full names of ACL users in Domino.

## Authority Propagation Mapping

The following chart documents the mapping between AS/400 object authorities and Notes.

| *AS/400 Object Authority* | *Notes Access Control List* |
|---|---|
| *objmgt, *objopr, *objexist, *objalter, *objref, *read, *upd, *dlt, *add (*ALL) | Manager |
| *objopr, *objalter, *objref, *read, *upd, *dlt, *add | Designer |
| *objopr, *read, *upd, *add, *dlt (*CHANGE) | Editor |
| *objopr, *read, *upd, *add | Editor with NoDelete |
| *objopr, *read, *upd | Editor with NoDelete (*) |
| *objopr, *read, *add, *dlt | Author |
| *objopr, *read, *add | Author with NoDelete |
| *objopr, *read, *dlt | Author with No create |
| *objopr, *read (*USE) | Reader |
| *objopr, *add, *upd | Depositor (**) |
| *objopr, *add | Depositor |
| Other combinations, (*EXCLUDE) | No Access |

***Note:** Authority level increased in Notes because there is no equivalency between AS/400 and Notes.

****Note:** Authority level decreased in Notes because this is no equivalency between AS/400 and Notes.

# Chapter 22
# Command Activity

This chapter provides information about the LEI Command Activity. Included is information about what this Activity is, when and why to use it, and how to create the Activity using the Command Activity document.

## Introduction to the Command Activity

A Command Activity executes operating system, database, and SQL commands.

In order to execute an operating system command, select No Connector and enter the command as it would be entered at an operating system command prompt.

To execute a database or Connection-specific command, select the desired Connection Document.

## When to Use a Command Activity

You should use the Command Activity when you need to execute a specific command either on a connected database or on the operating system on which the LEI server is running.

For example, you can use the Command Activity to:

- Execute an SQL statement on a supported Connection
- Execute an operating system command.

## How to Create a Command Activity

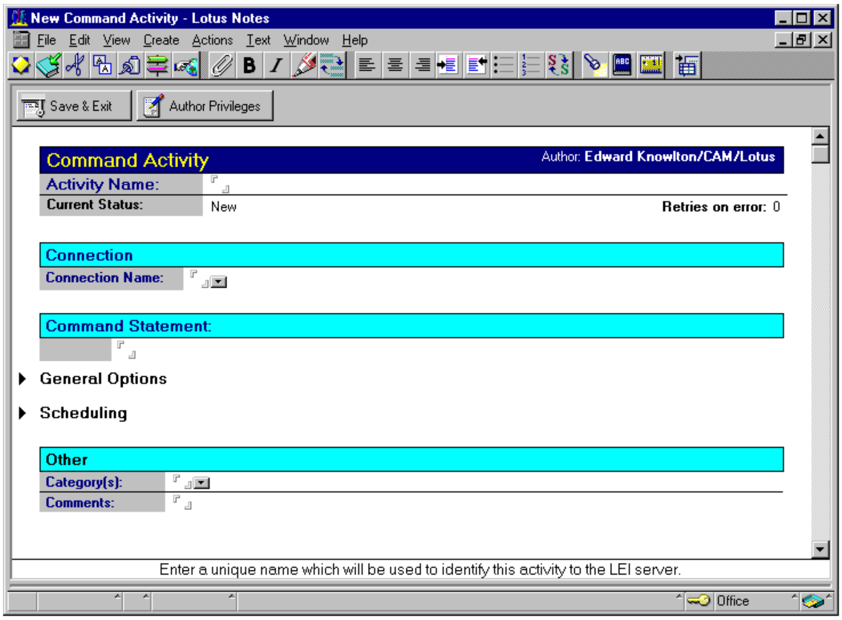
The steps below outline the general procedure for creating a Command Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – Command from the menu bar, or click Create Activity in the Navigator and then select Command.
3. Fill in the required fields and select the desired options for the Activity in the Command Activity document, including its Connections and Scheduling options. Note that any Connection you want to use in the Activity must already exist and should have been tested using the LCTEST described in Chapter 5, "Introduction to Connectors".
4. Save the Activity document.

You can select the Run ASAP option to have the Activity executed by the LEI Server as soon as possible after saving the Activity.

## Command Activity Document

Choose Create – Activity – Command from the menu bar or click Create Activity in the navigator to access the Command Activity document, shown below. The Command Activity document defines the information needed for a Command Activity. The fields and options in the document are described in the following sections.


New Command Activity - Lotus Notes

Command Activity Author: Edward Knowlton/CAM/Lotus

Activity Name:

Current Status: New Retries on error: 0

Connection

Connection Name:

Command Statement:

General Options

Scheduling

Other

Category(s):

Comments:

Enter a unique name which will be used to identify this activity to the LEI server.

See Chapter 17, “Introduction to Activities,” for a description of the following sections that are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

## Connection Properties

This section provides information on the Connections being used by the Activity. Connections must be created before they can be used in an Activity. For more information, see Chapter 5, "Introduction to Connectors," as well as the specific chapter for the type of Connection you are using in the Activity.

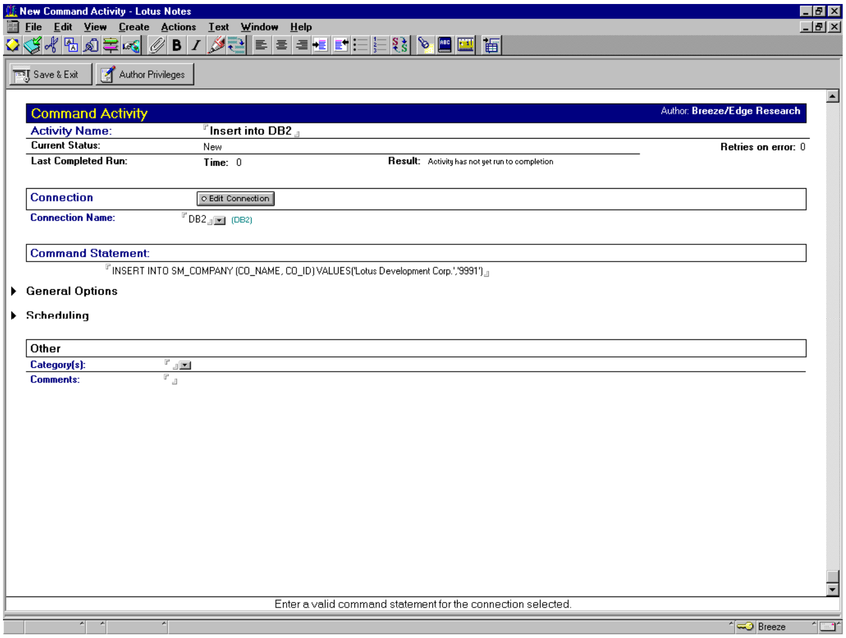
| *Field* | *Description* |
|---|---|
| Connection Name | Enter the name of the Connection. The command statement will be executed against this Connection. The Connection identifies the server and database along with required access information. To select a Connection, place the cursor in the edit field and press Enter to see the list of Connections. You can also place the cursor in the edit field and press a letter key to see the first Connection beginning with that letter. Repeatedly pressing a letter scrolls through all Connections with that initial letter. To execute a command in the local operating system, specify no Connection.<br><br>When using no Connection (for example when submitting an OS command), any command that takes input must have that input redirected from a file. If not, then the Activity will wait for input at the LEI console, and the Activity will seem to hang. So for example, while the command "del *.*" on NT will hang as it waits for confirmation, the command "del *.* < input.txt", where input.txt contains a "y" followed by a new line, will work fine, as the confirmation has been redirected to come from the file rather than from user input (stdin). |
| Edit Connection | The Edit Connection button (located in the blue title bar directly over the selected Connection's name) brings up the selected Connection's document in EDIT mode. |
| Command Statement | The Connector-specific (or operating-system specific, when no Connector is selected) statement that identifies the action to be performed. The statement must conform to the syntax rules of the database engine or local machine (for example, a SQL query or Notes selection formula). |

## Command Activity Examples

Following are some examples that illustrate how to use the Command Activity.

### Command Activity using SQL

This Command Activity inserts a row in a DB2 table. This example shows how it is possible to manipulate data in a relational database using an SQL statement within a Command Activity.


New Command Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Edit Connection
General Options
Scheduling
Breeze

Command Activity — Author: Breeze/Edge Research

| | | | |
|---|---|---|---|
| Activity Name: | Insert into DB2 | | |
| Current Status: | New | | Retries on error: 0 |
| Last Completed Run: | Time: 0 | Result: Activity has not yet run to completion | |

Connection

Connection Name: DB2 (DB2)

Command Statement:

```
INSERT INTO SM_COMPANY (CO_NAME, CO_ID) VALUES('Lotus Development Corp.','9991')
```

Other

Category(s):

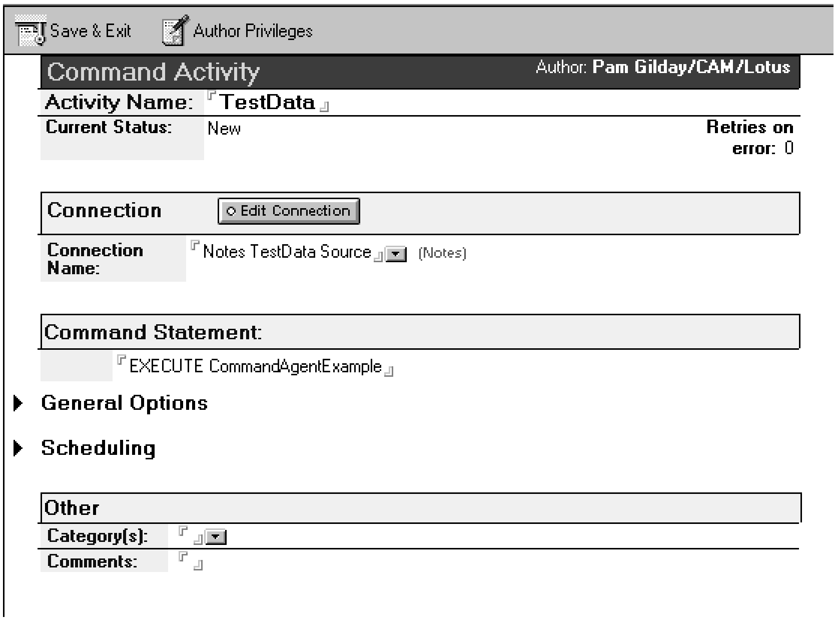
Comments:

Enter a valid command statement for the connection selected.

## Command Activity using a Notes Agent

This example runs a Notes Agent using the EXECUTE command statement. Notice that no quotes are required around the name of the Notes Agent.

The EXECUTE command statement is used to run a single Notes Agent. No command parameters are used.



Command Activity Author: Pam Gilday/CAM/Lotus

Activity Name: TestData

Current Status: New

Retries on error: 0

Connection

Connection Name: Notes TestData Source (Notes)

Command Statement:

EXECUTE CommandAgentExample

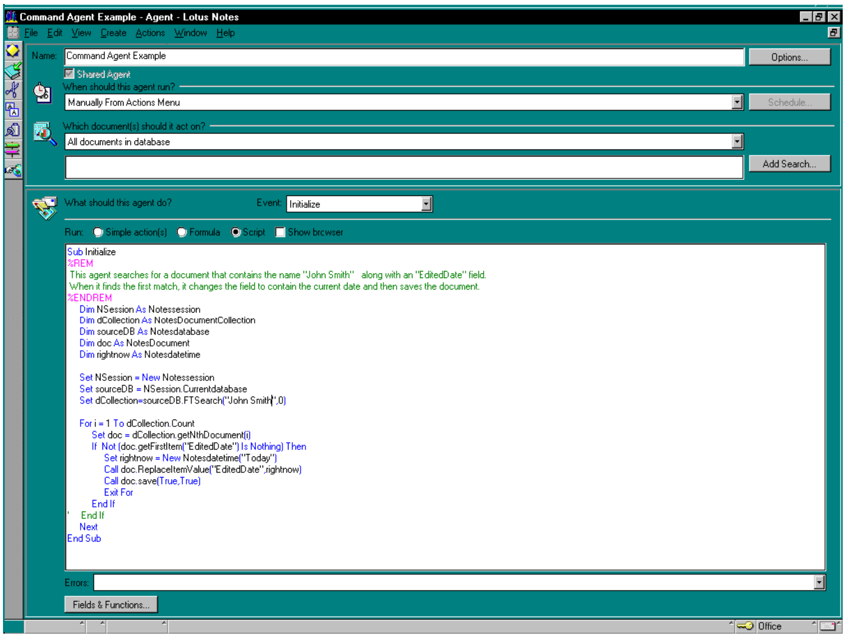
Other

Category(s):

Comments:

The agent shown below is executed by the above Command Activity.

This particular agent is designed to run against all documents in the database. The only other valid document selection option for an agent is “Run Once.” Agents cannot be view-based. The options “Run on selected documents in view” and “Run on all documents in view” are not supported. See Chapter 6, “Lotus Notes Connector”, for more information about Activity Command statements.


Command Agent Example - Agent - Lotus Notes
File Edit View Create Actions Window Help
Options...
Schedule...
Add Search...
Fields & Functions...
Office

Name: Command Agent Example

[✓ Shared Agent]

When should this agent run? Manually From Actions Menu

Which document(s) should it act on? All documents in database

What should this agent do? Event: Initialize

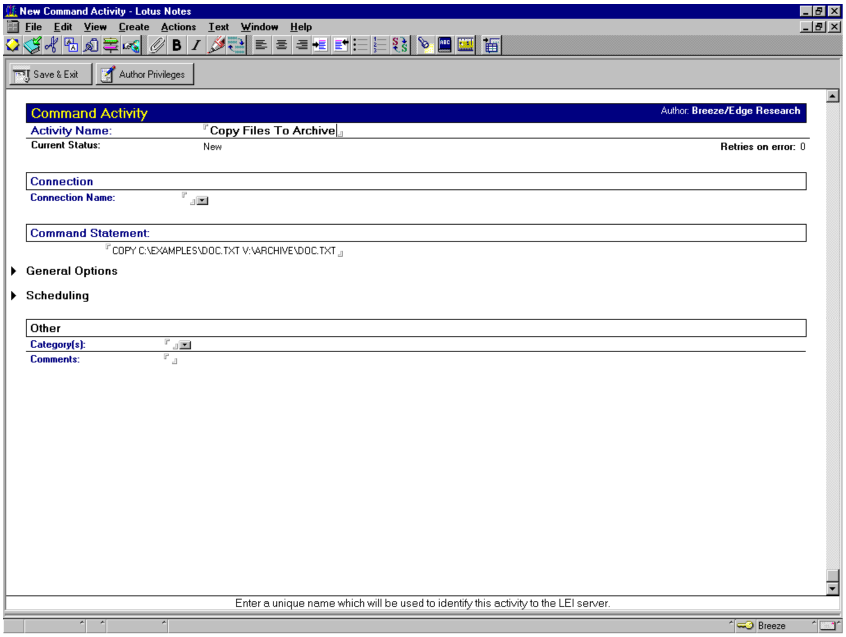
Run: Simple action(s) Formula [✓ Script] Show browser

```
Sub Initialize
%REM
 This agent searches for a document that contains the name "John Smith"   along with an "EditedDate" field.
 When it finds the first match, it changes the field to contain the current date and then saves the document.
%ENDREM
    Dim NSession As Notessession
    Dim dCollection As NotesDocumentCollection
    Dim sourceDB As Notesdatabase
    Dim doc As NotesDocument
    Dim rightnow As Notesdatetime

    Set NSession = New Notessession
    Set sourceDB = NSession.Currentdatabase
    Set dCollection=sourceDB.FTSearch("John Smith",0)

    For i = 1 To dCollection.Count
        Set doc = dCollection.getNthDocument(i)
        If  Not (doc.getFirstItem("EditedDate") Is Nothing) Then
            Set rightnow = New Notesdatetime("Today")
            Call doc.ReplaceItemValue("EditedDate",rightnow)
            Call doc.save(True,True)
            Exit For
        End If
'    End If
    Next
End Sub
```

Errors:

## Command Example Issuing Operating System Commands

This example issues an operating system specific command.


New Command Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges

Command Activity — Author: Breeze/Edge Research

Activity Name: Copy Files To Archive

Current Status: New — Retries on error: 0

Connection

Connection Name:

Command Statement:

```
COPY C:\EXAMPLES\DOC.TXT V:\ARCHIVE\DOC.TXT
```

General Options

Scheduling

Other

Category(s):

Comments:

Enter a unique name which will be used to identify this activity to the LEI server.

The Command Activity does not require that you specify a Connector in the Connection Name field of the Command Activity document. This capability allows you to enter command statements directed at the system. Here is another example of the kind of command this capability allows you to make:

```
del /q V:\SalesByMonth\June\*.*
```

You do not need to specify a File Connector to use this type of command statement.

**Caution** Inadvertently inserting a space in a file path can corrupt the Domino and LEI installations by deleting the lei.ini and notes.ini files. Be cautious when entering system commands.

# Chapter 23
# Direct Transfer Activity

This chapter provides information about the LEI Direct Transfer Activity. It includes information about this Activity, when to use it, and how to create the Activity.

## Introduction to the Direct Transfer Activity

A Direct Transfer transfers data from one database to another. The data to be transferred is identified by a statement, for instance, an SQL query or a Notes selection formula. Its execution can be immediate by clicking the Run ASAP button, scheduled via the Activity scheduling options, or by clicking the Start Activity icon in the Navigator.

## When to Use the Direct Transfer Activity

You should use the Direct Transfer Activity when you want to move data from one database to another.

You can combine a Direct Transfer Activity with a Polling Activity in order to do the data transfer when a specified condition is met.

## How to Create a Direct Transfer Activity

The steps below outline the general procedure for creating a Direct Transfer Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – Direct Transfer from the menu bar, or click Create Activity on the Navigator.
3. Fill in the required fields and select the desired options for the Activity in the Direct Transfer Activity document, including its Connections and Scheduling options.

**Note** All Connections that you want to use in the Activity must already exist and should already have been tested using the LCTEST described in Chapter 5, “Introduction to Connectors.”

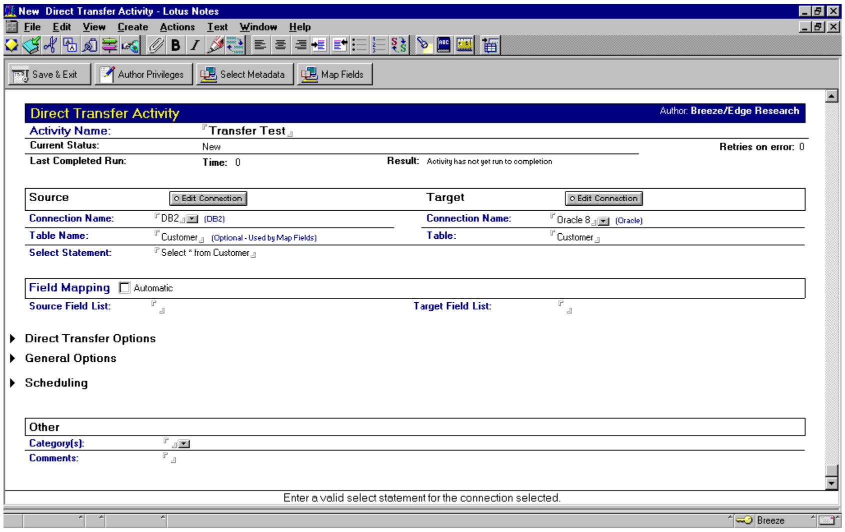
4. Save the Activity document.

You can select the Run ASAP option to have the Activity executed by the LEI Server as soon as possible after saving the Activity.

There is a complete example of building a Direct Transfer Activity in Appendix E, Tutorials.

## Direct Transfer Activity Document

Choose Create – Activity – Direct Transfer from the menu bar, or click Create Activity on the navigator to access the Direct Transfer Activity document, shown below. The Direct Transfer Activity document defines the information needed for a Direct Transfer Activity. The fields and options in the document are described below.



Direct Transfer Activity — Author: Breeze/Edge Research

| | | | |
|---|---|---|---|
| Activity Name: | Transfer Test | | |
| Current Status: | New | | Retries on error: 0 |
| Last Completed Run: | Time: 0 | Result: Activity has not yet run to completion | |

| | | | |
|---|---|---|---|
| Source | | Target | |
| Connection Name: | DB2 (DB2) | Connection Name: | Oracle 8 (Oracle) |
| Table Name: | Customer (Optional - Used by Map Fields) | Table: | Customer |
| Select Statement: | Select * from Customer | | |

Field Mapping ☐ Automatic

| | | | |
|---|---|---|---|
| Source Field List: | | Target Field List: | |

Direct Transfer Options

General Options

Scheduling

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

Enter a valid select statement for the connection selected.

See Chapter 17, “Introduction to LEI Activities,” for a description of the following sections that are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

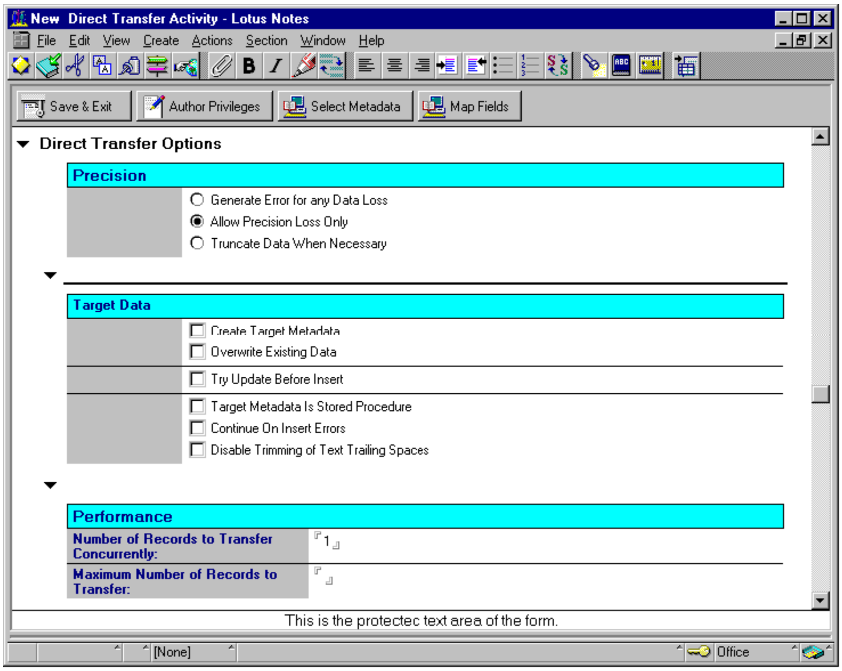
## Connection Properties

The Connection Properties section provides information on the data source and destination. Connectors are connections between the LEI Server and data sources and must be created before they can be used in an Activity document. For more information, see the chapter that describes the specific Connector that you are using.

| *Field* | *Description* |
|---|---|
| Edit Connection | The Edit Connection button (located in the blue title bar directly over the selected Connection's name) brings up the selected Connection's document in EDIT mode. |
| Connection Names | Names of the two Connections. The data is transferred from the Source Connection database to the Target Connection database. The Connection identifies the server and database with access information. The Additional Information segment of the document (see below) identifies the metadata.<br>To select a Connection, place the cursor in the edit field and press Enter to see the list of Connections. Repeatedly pressing a letter scrolls through all Connections with that initial letter. You can also place the cursor in the edit field and press a letter key to see the first Connection beginning with that letter. |
| Table Names | Visible only after you enter a name in the Connection Name field. Enter the table names of the Source and Target data you want to transfer. Click the Select Metadata button to see a list of table names. |
| Select Statement | Enter the command statement that identifies the data to be selected and transferred. The statement must conform to the syntax rules of the database engine (for example, a SQL query or Notes selection formula). A SQL query can be designed to select data from one or more metadata objects. |
| Field Mapping<br>• Source Field List<br>• Target Field List | If you would like to define your own field mappings, enter the names of the fields in the Source Field List and in the Target Field List in the proper mapping order. The first source field maps to the first target field, the second to the second, the third to the third, and so on.<br>For more information about Field Mapping, see Chapter 17, "Introduction to LEI Activities".<br>Automatic – If you mark Automatic, data is either mapped from each source column to the destination by position (first column to first, second to second, third to third, and so on) or by field name. |

## Direct Transfer Options

The Direct Transfer Options section of the document provides options for data handling during the Activity. Check any of the options that you want to use in this section. Each of the options is described in the table below.


New Direct Transfer Activity - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields

Direct Transfer Options

Precision

- ○ Generate Error for any Data Loss
- ● Allow Precision Loss Only
- ○ Truncate Data When Necessary

Target Data

- ☐ Create Target Metadata
- ☐ Overwrite Existing Data
- ☐ Try Update Before Insert
- ☐ Target Metadata Is Stored Procedure
- ☐ Continue On Insert Errors
- ☐ Disable Trimming of Text Trailing Spaces

Performance

| | |
|---|---|
| Number of Records to Transfer Concurrently: | 1 |
| Maximum Number of Records to Transfer: | |

This is the protectec text area of the form.

[None] Office

| *Field* | *Description* |
|---|---|
| Create Metadata If Necessary | LEI will create a new metadata in the database identified in the Target Connection to hold the data being transferred if one does not already exist. In creating a new metadata, LEI will try to match the source metadata as much as possible.<br><br>If the source table contains no records, the Direct Transfer Activity will not transfer the table. To transfer the table, in cases where all table records contain NULL entries, add one record to the table and then perform the Direct Transfer Activity. |
| Overwrite Existing Data | Checking this box causes all the data in the destination metadata to be deleted before the new data is transferred. Note that even if the transfer fails to complete successfully, the data already has been deleted from the target and will be lost.<br><br>If this option is not checked, the transferred data is appended to the existing data in the target. |
| Destination Metadata Is Stored Procedure | When set, the target metadata indicates a stored procedure to execute. The fields being stored become parameters to the stored procedure with the field names as parameter names. |
| Continue on Insert Errors | Enable this option to have the Activity log any insertion errors and continue with the transfer. |
| Disable Trimming of Text Trailing Spaces | By default, LEI trims trailing spaces from text fields. Select this option to cause LEI to keep any trailing spaces in text fields. |
| Generate Error for any Data Loss | LEI writes an error to the log for any data that's lost as a result of the transfer and terminates the transfer. |
| Allow Precision Loss Only | LEI does not report loss of numerical or datetime precision as a result of the transfer. This option is the default. |

*continued*

| *Field* | *Description* |
|---|---|
| Truncate Data When Necessary | LEI truncates text data when necessary to conform to field lengths in the target database. Because of performance penalties, the truncate data option does not affect numbers being written. |
| Try Update Before Insert | When selected, this option prompts for a keyfield to use to check for records to update. The keyfield is used to find the data record in the target, and, if found, the record will be “updated” (whether it changed or not) on the destination. If a record from the source (based on keyfield), does not match up with a record in the target, then that record will be “inserted” into the target as a new record. |
| Number of Records to Transfer Concurrently | By default, one record is transferred at a time. For databases and Connections that support array transfers, this number defines the number of records to transfer at a time. Transferring multiple records at a time can increase performance, but only with databases that support the feature. |
| Maximum Number of Records to Transfer | Enter the total number of records that you want this Activity to transfer. 0 equals no limit. |

# Chapter 24
# Polling Activity

This chapter provides information about the LEI Polling Activity. Included is information about the Activity, when to use it, and how to create it.

## Introduction to the Polling Activity

Polling is a way of causing an Activity (such as a Direct Transfer or Replication) to execute when a condition is met. The Polling document itself does not contain the instructions to execute. Instead, it triggers another Activity or activities that are configured in separate documents.

The Polling Activity does not reestablish a closed Connection to a data source. If you are Polling a Connection, the Connection must remain open as long as necessary relative to the Polling interval. For example, if you have configured Sybase to time out its Connection after 10 minutes of inactivity, but your Polling Activity has a polling interval set to 15 minutes, the Sybase Connection may time out and close before the Polling Activity checks the Connection. The Polling Activity would fail in this case.

## When to Use the Polling Activity

You should use the Polling Activity when you need to execute some action when another action or event occurs. For example, say the Human Resources department uses an Oracle database for employee information, and you need to maintain this same information in a Notes database so the Engineering, Sales and Marketing departments can access it. You could use a Polling Activity to check the Oracle database for new or changed records, and when the Polling Activity detects such an event, it would trigger a Replication Activity to copy the changed data to the Notes database.

## How to Create a Polling Activity

The Polling Activity document contains the information needed to execute subordinate activities, including:

- The condition under which the subordinate activities (the activities to execute) will be run.
- Scheduling and polling frequency.
- Activities to be executed.

The polled database, including access information, must be defined in a Connection document before the Polling Activity is created.

Subordinate activities must also be defined (in Activity documents) before you create a Polling Activity.

The steps below outline the general procedure for creating a Polling Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

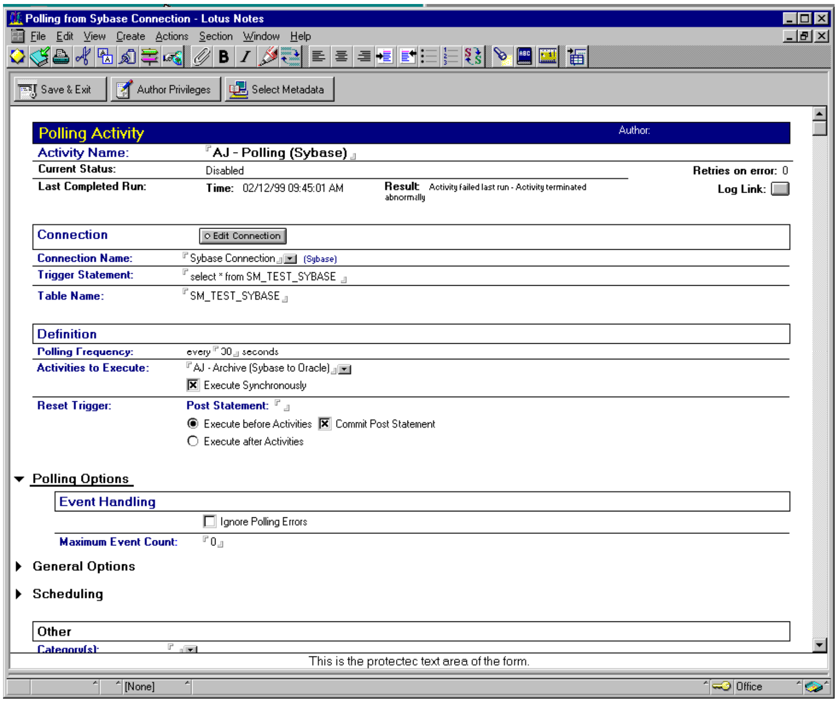
1. Open the LEI Administrator database, leiadm.nsf.
2. Click Create Activity on the Navigator and select Polling, or choose Create – Activity – Polling from the menu.
3. Fill in the required fields and select the desired options for the Activity in the Polling Activity document, including its Connections and Scheduling options. Note that any Connections you want to use in the Activity must already exist and should already have been tested using the LCTEST described in Chapter 5, "Introduction to Connectors."
4. Save the Activity document.

   You can select the Run ASAP option to have the Activity executed by the LEI server as soon as possible after saving the Activity.

There is a complete example of building a Polling Activity in Appendix E, Tutorials.

## The Polling Activity Document

Choose Create – Activity – Polling from the menu bar, or click Create Activity on the navigator and select Polling to access the Polling Activity document, shown below. The Polling Activity document defines the information needed for the Activity. The fields and options in the document are described below.


Polling from Sybase Connection - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata

Polling Activity — Author:

Activity Name: AJ - Polling (Sybase)

Current Status: Disabled — Retries on error: 0

Last Completed Run: Time: 02/12/99 09:45:01 AM — Result: Activity failed last run - Activity terminated abnormally — Log Link:

Connection — Edit Connection

Connection Name: Sybase Connection (Sybase)

Trigger Statement: select * from SM_TEST_SYBASE

Table Name: SM_TEST_SYBASE

Definition

Polling Frequency: every 30 seconds

Activities to Execute: AJ - Archive (Sybase to Oracle)

[✓] Execute Synchronously

Reset Trigger: Post Statement:

● Execute before Activities [✓] Commit Post Statement

○ Execute after Activities

Polling Options

Event Handling

[ ] Ignore Polling Errors

Maximum Event Count: 0

General Options

Scheduling

Other

Category(s):

This is the protectec text area of the form.

[None] — Office

See Chapter 17, “Introduction to LEI Activities,” for descriptions of the following sections of the Polling Activity document that are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

## Connection Fields

This section provides information on the source of the data to be polled. Connections must be created before they can be used in a Polling Activity document.

| *Field* | *Description* |
|---|---|
| Edit Connection | The Edit Connection button (located in the blue title bar directly over the selected Connection's name) is available only after you enter a Connection name. Clicking this button brings up the selected Connection's document in EDIT mode. |
| Connection Name | Enter the name of the Connection containing the database to be polled.<br><br>To select the Connection, place the cursor in the edit field and press Enter to see the list of Connections. You can also place the cursor in the edit field and press a letter key to see the first Connector beginning with that letter. Repeatedly pressing the space bar scrolls through all Connectors.<br><br>After selecting the Connection, you can click the Select Metadata action button to browse the associated database for metadata and field information. |
| Trigger Statement | Enter the Connection-specific trigger statement that you want to execute. This statement identifies the condition that will cause the Polling Activity to place the activities to execute in a queue for execution. The condition is satisfied when the statement produces a result set containing at least one record. This is considered a successful poll. |
| Table/Form/Subdirectory Name | The metadata for trigger, based on the type of Connection chosen. For example, in a relational database the metadata is a Table, in a Notes database the metadata is a Form, in a File the metadata is a subdirectory. |

## Definition Fields

This table provides information about the Definition fields in the Polling Activity document.

| *Field* | *Description* |
|---|---|
| Polling Frequency | Specify, in number of seconds, how often the database is to be polled to check for the condition defined in the command statement. The Polling only takes place within the window of time defined in Scheduling. See "Notes on Polling Activity Scheduling" at the end of this chapter. |
| Activities to Execute | The list of activities to execute when result of the command statement formula is true. Place the cursor in the edit field and press Enter to select activities.<br>Listing an Activity here will have no effect on its scheduling or whether it can be run ASAP. To prevent an Activity from running on a schedule, make sure that its schedule is disabled. |
| Execute Synchronously | By default, Polling is not suspended while an Activity takes place. If this option is checked Polling is suspended while triggered activities are executed.<br>Because synchronous polling is suspended until the triggered Activity is completed, polling will not resume if the triggered Activity cannot complete for any reason (for example, the server that was supposed to run it is not up). In this case, the Polling Activity will remain suspended until closed. |
| Reset Trigger | Options regulating a statement executed after a successful poll has occurred. |
| Post Statement | The optional statement to execute before or after a successful poll. This statement can be used to reset the condition that caused the poll to succeed, for example. You can choose to have the statement execute before or after the activities to execute.<br>For Notes, the statement can run an agent using the syntax Execute XXX. The agent must run on "All documents in database," a setting applied when designing the agent. |
| Execute before Activities<br>Execute after Activities | When statements are set to run after the activities execute, set the synchronous flag to ensure that the executed activities have finished before the post Activity statement begins. |
| Commit Post Statement | For Connections which support a Commit Action, select this check box to explicitly commit the post-statement following each execution. |

## Polling Options Fields

This table provides information about the Polling fields in the Polling Activity document.

| *Field* | *Description* |
|---|---|
| Ignore Polling Errors | Check this option to continue polling if the polling statement returns an error. This can be used as a retry option: wait until an event occurs, ignoring interim errors, and then perform an action. |
| Maximum Event Count | The number of times the Activity will poll successfully before terminating all polling. A value of 0 means that there's no limit to the number of times polling can occur. |

## Notes on Polling Activity Scheduling

- Dependent activities (those launched by a Polling Activity) are executed when the process of polling stops and have no connection with polling frequency or with activities executed as a result of polling conditions.
- Polling Frequency and Maximum Event Count can be used in connection with Scheduling to define the polling period.
- Unlike other activities that continue their current run past any defined Stop Time until they complete, a Polling Activity, once started, runs up to Stop Time and ceases.
- When a Polling Activity is run via the Administrator, it recalculates its next invalid run time (that is, its Stop time). However, when a Polling Activity is run via shared execution, the Stop time is not recalculated. When a Polling Activity is run through the CGI or from the command line, it is not rescheduled and the invalid (Stop) value is not updated.
- A Polling Activity will attempt to reconnect a dropped connection, but if the server has gone down or is not immediately available, the Activity will not make any further attempt at reconnection.

# Chapter 25
# RealTime Notes Activity

This chapter provides information about the LEI RealTime Notes Activity. Included is information about this Activity, suggestions on when to use it, examples of usage, and how to set up and create a RealTime Notes Activity.

## Introduction to the RealTime Notes Activity

A RealTime Notes Activity provides synchronous access from a Notes application to an LEI-supported external data source.

The LEI RealTime Notes Activity intercepts Notes database events and acts on them. When Notes users open, create, update, or save Notes documents, these events are intercepted and acted upon, obtaining "real-time" access from the Notes form to back-end sources supported by the LEI server. With the RealTime Notes Activity, Notes end-users can open, create, update or delete back-end data directly and transparently through their familiar Notes Clients. By extension, Web clients may open up the same Notes forms by accessing a Domino server (Releases 4.6 .x or 5.0.x or greater) and also obtain RealTime access to LEI supported back-end source data.

The Notes database events are intercepted and handled by the LEI server hosted on the Domino server that owns the Notes application.

For example, if the external database to be queried or updated from the Notes form is DB2, Notes end-users may work with DB2 data as if it were in Notes. In addition, DB2 connectivity software is not required on the client system. Network access to the back end source is handled by the Domino and LEI server machine, which contains the connectivity software for the database. No additional coding is necessary in the Notes application. In addition, it is possible to control which fields are actually stored in the Notes database, significantly reducing storage requirements on the Notes side.

Several items are required to create a RealTime Notes Activity from within the LEI Administrator. Each RealTime Notes Activity monitors a specific Notes database and requires a Notes Form to define the metadata. A single external Connection indicates the external metadata (usually an SQL table) to use. Key and data field mappings are also required. Several RealTime Notes Activities can be used to monitor different databases, a single database or even a single form. This means a single document can be populated real-time, consisting of data from multiple back-end databases using a RealTime Notes Activity for each of the various back-end data sources.

### Requirements

Use of the RealTime Notes Activity requires that the LEI server be installed on the Domino server (Releases 4.6 .x or 5.0.x or greater) hosting the Notes database forms that are to be monitored.

## About RealTime Notes Activity Events

The following types of Notes database events can be intercepted, causing the LEI server to query, create, update, or delete a record in an LEI supported external data source. These events occur against "stub" documents in the Notes application. These stub documents contain, at a minimum, a key field or fields that exist in common between the Notes form and the back-end metadata. The Notes fields mapped to the external Connection will be populated or used to populate external records depending on which of the following events the RealTime Notes Activity tracks.

**Open** – When a Notes document is opened, data is retrieved from the external Connection. The key fields, which are always stored in the Notes document, are used to query the corresponding record in the external Connection, and the data fields are transferred into the Notes document.

**Update** – When a Notes document is modified, the changed data fields are propagated to the external Connection, updating the corresponding external record. (See the Data Storage options in the RealTime Notes Options section below.)

**Create** – When a new Notes document is saved, a corresponding record is created in the external Connection. (See the Data Storage options in the RealTime Notes Options section below.)

**Delete** – When a Notes document is deleted, the key fields are used to delete the corresponding record in the external Connection. If a delete event cannot locate the corresponding record in the external system, no error is returned since the intended effect (no document with the corresponding keys) has been achieved. (See the RealTime Notes Options section below).

**Note** Records inserted into the external data through another front end are not represented in the Notes database by stub documents unless they are explicitly transferred. One way to do this is to shut down the RealTime Notes Activity, periodically run a Replication (which only includes key fields and only propagates inserts and optional deletions) from the external Connection to Notes, and then restart the RealTime Notes Activity.

## When to Use the RealTime Notes Activity

You should use a RealTime Notes Activity when you want to provide back-end data to Notes client or Domino Web browser users without storing duplicate data in the Notes database other than the key fields.

**Note** LEI does NOT support a Notes to Notes RealTime Activity. Specifically, in a Notes RealTime Activity, a Notes database cannot be the back-end.

## How to Create a RealTime Notes Activity

This section describes how to create a RealTime Notes Activity. For a step-by-step example of this process, see Appendix E.

### Preparing to Create a RealTime Notes Activity

The following items may be helpful in preparing to create a RealTime Notes Activity:

1. Create a form in a Notes database that is located on the Domino server hosting LEI that contains fields for the keys and data you want to use from the external database.
2. Create a Connection document to the Notes database above and to the external database.
3. Create a Direct Transfer Activity that transfers the key field or fields from the external database to the Notes database form created in Step 1. You may later want to set up a Replication Activity to keep the keys in the external database in sync with the keys stored in the "stub" documents in Notes. This is suggested if data is being entered directly into the external database through another front end.

### Creating a RealTime Notes Activity

The steps below outline the general procedure for creating a RealTime Notes Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – RealTime Notes from the menu bar, or click Create Activity on the Navigator and select RealTime Notes.

3. Fill in the required fields and select the desired options in the RealTime Notes Activity document, including Connection and Scheduling options.

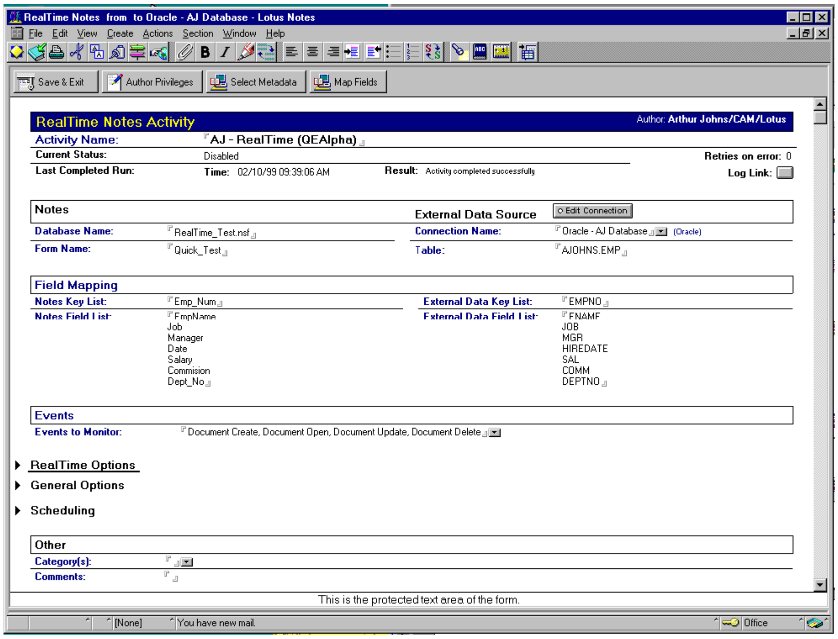
   **Note** All Connections that you want to use in the Activity must already exist and should have been tested using the LCTEST described in Chapter 5, "Introduction to Connectors."

4. Save the Activity document.
5. After you have saved the RealTime Notes Activity you must run it to make it active.

Refer to Appendix E, "Tutorials" for a tutorial and sample database that you can use to create a RealTime Notes Activity.

## RealTime Notes Activity Document

Choose Create – Activity – RealTime Notes from the menu bar, or click Create Activity on the Navigator and select RealTime Notes to access the RealTime Notes Activity document, shown below. The RealTime Notes Activity document defines the information needed for this Activity. The fields and options in the document are described below.


RealTime Notes from to Oracle - AJ Database - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields
Edit Connection
[None]
You have new mail.
Office

RealTime Notes Activity — Author: Arthur Johns/CAM/Lotus

Activity Name: AJ - RealTime (QEAlpha)
Current Status: Disabled — Retries on error: 0
Last Completed Run: Time: 02/10/99 09:39:06 AM — Result: Activity completed successfully — Log Link:

| Notes | | External Data Source | |
|---|---|---|---|
| Database Name: | RealTime_Test.nsf | Connection Name: | Oracle - AJ Database (Oracle) |
| Form Name: | Quick_Test | Table: | AJOHNS.EMP |

Field Mapping

| | | | |
|---|---|---|---|
| Notes Key List: | Emp_Num | External Data Key List: | EMPNO |
| Notes Field List: | EmpName<br>Job<br>Manager<br>Date<br>Salary<br>Commision<br>Dept_No | External Data Field List: | FNAME<br>JOB<br>MGR<br>HIREDATE<br>SAL<br>COMM<br>DEPTNO |

Events

Events to Monitor: Document Create, Document Open, Document Update, Document Delete

- RealTime Options
- General Options
- Scheduling

Other

Category(s):

Comments:

This is the protected text area of the form.

See Chapter 17, "Introduction to LEI Activities", for descriptions of the following sections, which are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

## Notes and External Data Source

The Notes and External Data Source Connections section of the RealTime Notes Activity document provides information on the monitored database and External Data Source Connection.

| *Field* | *Description* |
|---|---|
| Notes Database Name | The relative file path and name of the local Notes database. |
| Notes Form Name | Name of the Notes form to monitor. |
| Edit Connection | The Edit Connection button (located in the blue title bar directly over the selected Connection's name) brings up the selected Connection's document in EDIT mode. |
| External Data Source Connection Name | Name of the External Data Source Connection to use.<br><br>To select a Connection, place the cursor in the edit field and press Enter to see the list of Connectors. You can also place the cursor in the edit field and press a letter key to see the first Connection beginning with that letter. Repeatedly pressing a letter scrolls through all Connections with that initial letter. |
| External Data Source Table Name | The metadata name of the external data source. |

## Field Mapping

The Field Mapping section defines the field mapping between the Notes form and the external database. See the table below for more information.

| *Field* | *Description* |
|---|---|
| Notes Key/External Data Key List | Enter the primary key or keys in this field. The key is used to locate the corresponding record in the External Connection. At least one key field is required for both the Notes and the external database. Corresponding key fields must be mapped to each other. You can click the Select Metadata action button to aid you in entering keys. |
| Notes Field/External Data Field List | Enter the names of the data fields in the source and destination database in the proper mapping order. The first Notes field maps to the first External Connection field, the second to the second, and so on. You can click the Map Fields action button to aid you in entering data fields.<br>You will receive an error if you enter the key field in this list. |

## Events

Use the Events section of the form to choose the Notes document events that you want the RealTime Notes Activity to intercept and process. See the table below for a description of the options.

| *Events to Monitor* | *Description* |
|---|---|
| Document Create | Handles new document creations on save and propagates the new record to the external Connection. |
| Document Open | Handles document opens and retrieves corresponding data from the external Connection. |
| Document Update | Handles updates on save and propagates changes to the external connection. |
| Document Delete | Handles document deletions and removes the corresponding record from the external connection. |

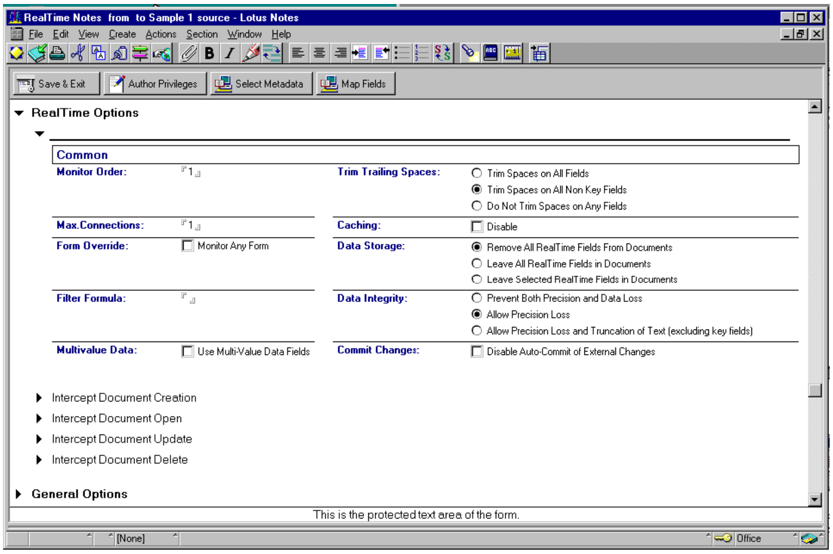
## RealTime Options

The RealTime Notes Options section of the document, shown below, includes five sections of options:

- Common Options
- Intercept Document Creation
- Intercept Document Open
- Intercept Document Update
- Intercept Document Delete

### Common Options

The Common Options section of the RealTime Notes Activity document, shown below, provides options that can be applied to the RealTime Notes Activity regardless of the type of Event the Activity monitors. Each option is described in the table below.


RealTime Notes from to Sample 1 source - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields

RealTime Options

Common

| | | | |
|---|---|---|---|
| Monitor Order: | 1 | Trim Trailing Spaces: | ○ Trim Spaces on All Fields<br>● Trim Spaces on All Non Key Fields<br>○ Do Not Trim Spaces on Any Fields |
| Max.Connections: | 1 | Caching: | ☐ Disable |
| Form Override: | ☐ Monitor Any Form | Data Storage: | ● Remove All RealTime Fields From Documents<br>○ Leave All RealTime Fields in Documents<br>○ Leave Selected RealTime Fields in Documents |
| Filter Formula: | | Data Integrity: | ○ Prevent Both Precision and Data Loss<br>● Allow Precision Loss<br>○ Allow Precision Loss and Truncation of Text (excluding key fields) |
| Multivalue Data: | ☐ Use Multi-Value Data Fields | Commit Changes: | ☐ Disable Auto-Commit of External Changes |

Intercept Document Creation
Intercept Document Open
Intercept Document Update
Intercept Document Delete

General Options

This is the protected text area of the form.

| *Option* | *Description* |
|---|---|
| Monitor Order | If you use more than one RealTime Notes Activity for a single Notes form, you may specify the order the Activities will intercept the document's events.<br><br>The monitor order also enables you to use multiple RealTime Activities which connect to different tables, and use values found by the first Activity (monitor order 1) as keys for subsequent RealTime Activities (monitor order 2, 3, etc.).<br><br>**Note** When adding or updating data to a source from a document that is not the first in the monitor order, a row is created in the source which may not contain all field or key values. To include all data and key fields, use one of the following methods:<br>• For each RealTime Activity preceding the last RealTime Activity, select the Data Storage option Leave All RealTime Fields in Document. For the last RealTime Activity, indicate to leave specific fields in the document, and list all key fields from the other RealTime Activities.<br>• An alternate method is to indicate on each RealTime Activity to leave only those fields in the document that will be referenced by subsequent RealTime Activities. |
| Max. Connections | This option sets the maximum number of Connections to the external database that can be open to service concurrent user requests simultaneously. LEI opens one Connection to the external data source when the first Notes application event occurs. If two or more events occur simultaneously, additional Connections are made, up to the maximum number of Connections specified by this option. When the maximum number of Connections is reached, subsequent events are queued and occur when each preceding event is serviced. Each event lasts only as long as necessary to read from or write data to the external data source. While the Connection is persistent, the time required to service each event is minimal, depending on the amount of data being read or written. The maximum number of Connections, therefore, does not need to be that great in order to service multiple events. We recommend that you set the maximum Connections to 2 or 3, and if users experience significant delays, you can increase this number. |
| Form Override | By default, only documents created with the form named in the Form Name field in the "Notes" section of the document are monitored. Select this option to monitor all documents in the database, regardless of their form. In this case, the metadata for the Notes Form is still used for key field mapping, data field mapping and type-checking. |

*continued*

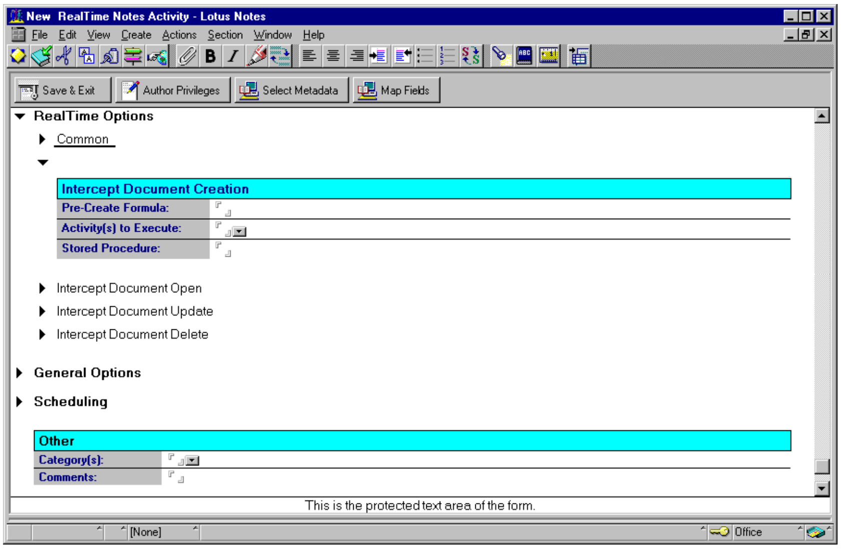
| *Option* | *Description* |
|---|---|
| Filter Formula | An optional Notes formula defining the documents that the Activity will monitor. Use this option to cause the RealTime Notes Activity to process only documents that satisfy the specified formula.<br><br>For all events, you can only reference fields in the Notes document. |
| Multivalue Data | Enable this option to have LEI consider non-key fields to be multi-value fields. If you use the Collapse/Expand MetaConnection, you must select this option. If you select this option, you must use the Collapse/Expand MetaConnection. |
| Trim Trailing Spaces | This option affects any trailing spaces that exist in the Text fields of the external data.<br><br>Text trimming only occurs when the fields are read from the external source.<br><br>There are three choices:<br>• Trim spaces on all fields – Select this setting to trim trailing spaces from all text fields.<br>• Trim spaces on all non key fields – Select this setting to trim trailing spaces only from data fields, not from the key fields. (Default)<br>• Do not trim spaces on any fields – Select this option to leave trailing spaces in all fields.<br><br>**Note** Trailing spaces may be required to ensure matching of fields between Notes and the external data.<br><br>**Caution** When data is put into backend fields of fixed length and then retrieved, the Connection with the backend may be lost because the backend has padded the data with spaces to fit the fixed length of the field. If this occurs, use either a variable length field (VARCHAR) in the backend database or enable the “Trim spaces on all non key fields” option. You will most likely encounter this situation when you use data of type CHAR as a key field with Oracle backends. When you use a CHAR data type as a key field, Lotus recommends that you use the default setting, “Trim spaces on non-key fields.” |
| Caching | Select this option to disable caching in the HTTP server for monitored documents. When a document is retrieved, the HTTP server in Domino may cache it to avoid disk access for the next retrieval. For occasionally changing external records, caching may be fine (for example, an employee handbook); for a RealTime situation with continuously changing data (for example, checking on an order status), caching should be disabled. |

*continued*

| *Option* | *Description* |
|---|---|
| Data Storage | Remove All RealTime Fields from Documents – Enable this option if you want to remove all the data fields mapped in the Activity from the Notes document before it is saved to disk. This is the default.<br><br>Leave All RealTime Fields in Documents – Enable this option if you want to leave all the data fields in the Notes document, rather than removing them after updating the external source (see above). This option only takes effect when creating or updating a document when the Activity is active.<br><br>Leave Selected RealTime Fields in Documents – Enable this option if you want to leave selected data fields in the Notes document. You may want to select this option in order to enable views of the forms. The button that appears when you enable this option lists the Notes data fields from which you can select.<br><br>**Note** Not selecting any fields is equivalent to enabling the Remove All RealTime Fields from Documents option. |
| Data Integrity | Prevent both precision and data loss – Enable this option to have LEI write an error to the log for any data that's lost as a result of the transfer and terminate the transfer.<br><br>Allow precision loss – Enable this option to have LEI not report loss of numerical or datetime precision as a result of the transfer. This allows some loss of precision without stopping the transfer. This option is the default.<br><br>Allow precision loss and truncation of text (excluding key fields) – Enable this option to have LEI allow precision loss and to truncate text data when necessary to conform to field lengths in the external database.<br><br>**Note** Key fields are not truncated. |
| Commit Changes | Disable Auto-commit of External Changes – By default, LEI automatically commits all data modification operations (insert, update, and delete). If changes are not automatically committed, then modifications may lock an external record or the entire table, deadlocking other users. Only disable Auto-commit if the external database auto-commits by default, or when using an alternative method of enabling Auto-commit (such as Connector options – see a specific Connector chapter for more information). |

## Intercept Document Creation Options

The Intercept Document Creation Options section of the RealTime Notes Activity document is shown below. The Intercept Document Creation section handles new document creations on save and propagates the new record to the external Connection.


New RealTime Notes Activity - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields
RealTime Options
Common
Intercept Document Open
Intercept Document Update
Intercept Document Delete
General Options
Scheduling
[None]
Office

Intercept Document Creation

Pre-Create Formula:

Activity(s) to Execute:

Stored Procedure:

Other

Category(s):

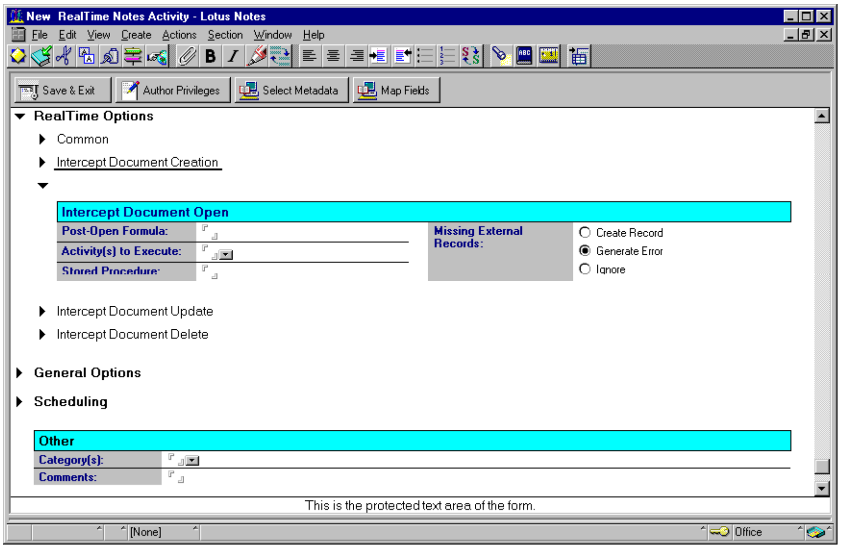
Comments:

This is the protected text area of the form.

| *Intercept Document Create Field* | *Description* |
|---|---|
| Pre-create Formula | Notes formula language statement to execute on the new Notes document prior to the creation of a new record in the external database. For example: `FIELD LASTNAME:=@if(LASTNAME = "";"NA",LASTNAME);""` **Note** If you are connecting to an Oracle backend and you must use text fields of fixed length, use a formula filter to pad the specific field to the correct length. See the Oracle Connector documentation for more information. |
| Activity(s) to Execute | Activity(s) to execute as part of the create event. These are LEI Activities defined in the administrator, and are executed immediately on the local LEI server regardless of Activity settings. |
| Stored Procedure to Execute | Stored Procedure to execute in place of normal create operation (input parameters are key and data values). |

## Intercept Document Open Options

The Intercept Document Open Options section of the RealTime Notes Activity document is shown below. The Intercept Document Open Options section handles document opens and retrieves corresponding data from the external Connection.


New RealTime Notes Activity - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields

RealTime Options

Common

Intercept Document Creation

Intercept Document Open

Post-Open Formula:

Activity(s) to Execute:

Stored Procedure:

Missing External Records: Create Record [✓ Generate Error] Ignore

Intercept Document Update

Intercept Document Delete

General Options

Scheduling

Other

Category(s):

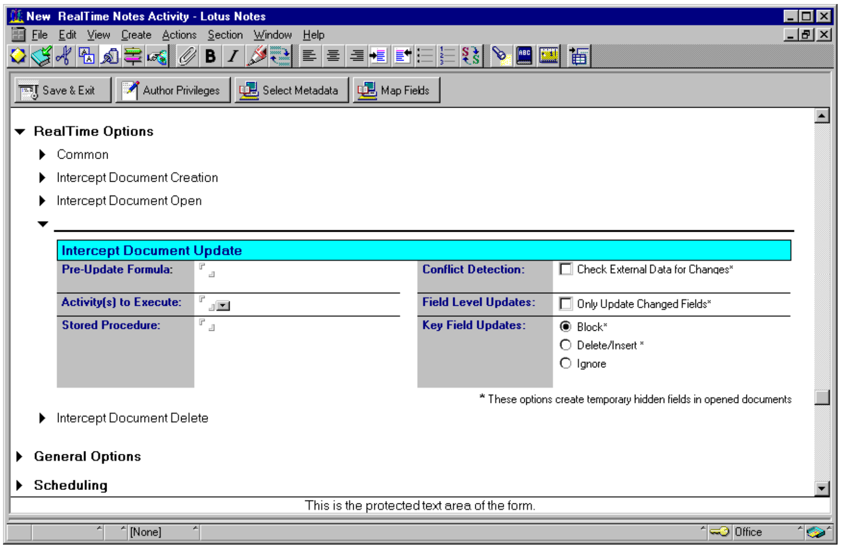
Comments:

This is the protected text area of the form.

| *Intercept Document Open Field* | *Description* |
|---|---|
| Post-Open Formula | Notes formula language statement to execute on the Notes document immediately following the retrieval of the external data. |
| Activity(s) to Execute | Activity(s) to execute as part of the Open event. These are LEI Activities defined in the Administrator and are executed immediately regardless of Activity schedule settings. |
| Stored Procedure | Stored Procedure to execute in place of normal select operation (input parameters are key values; output parameters are key and data values). |
| Missing External Records | Specifies how to record instances of missing records. |

## Intercept Document Update Options

The Intercept Document Update Options section of the RealTime Notes Activity document is shown below. The Intercept Document Update Options section handles updates to existing Notes documents and propagates changes to the external Connection.


New RealTime Notes Activity - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields
RealTime Options
Common
Intercept Document Creation
Intercept Document Open
Intercept Document Update

| | | | |
|---|---|---|---|
| Pre-Update Formula: | | Conflict Detection: | ☐ Check External Data for Changes* |
| Activity(s) to Execute: | | Field Level Updates: | ☐ Only Update Changed Fields* |
| Stored Procedure: | | Key Field Updates: | ● Block* ○ Delete/Insert * ○ Ignore |

* These options create temporary hidden fields in opened documents

Intercept Document Delete
General Options
Scheduling
This is the protected text area of the form.
[None]
Office

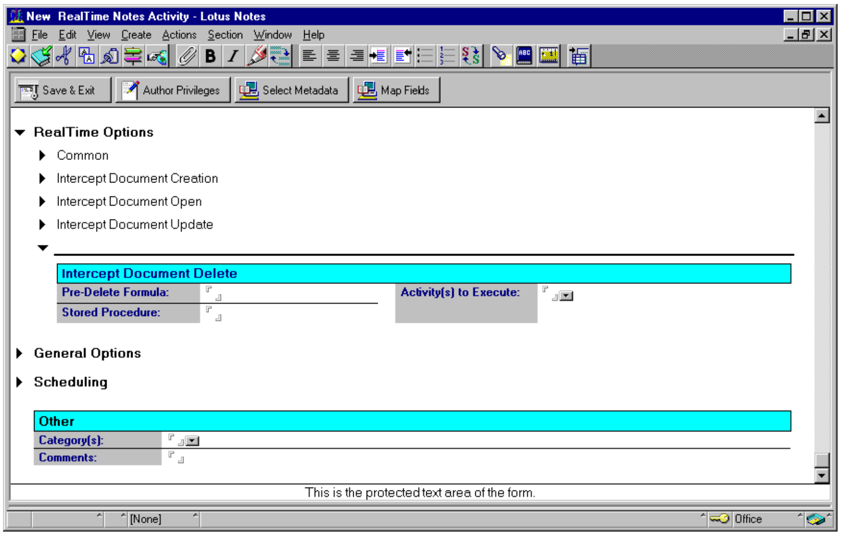
| *Intercept Document Update Field* | *Description* |
|---|---|
| Pre-Update Formula | Notes formula language statement to execute on the Notes document prior to the update in the external database. |
| Activity(s) to Execute | Activity(s) to execute as part of the update event are executed immediately, regardless of Activity schedule settings. These are LEI Activities defined in the Administrator. |
| Stored Procedure | Stored Procedure to execute in place of normal update operation (input parameters are key and data values). |
| Conflict Detection | Checking this option ensures that the Connection data has not changed since the document was open. If there were changes, the update to the Connection will fail. |

*continued*

| *Intercept Document Update Field* | *Description* |
|---|---|
| Field Level Updates | Checking this option causes a RealTime Notes Activity to not update fields in the Connection unless the corresponding fields in the Notes document have been edited. For example, use this option for monitoring updates to individual columns of a table. |
| Key Field Updates | • Block – Do not allow updates to key fields in the Notes document or the Connection records.<br>• Delete/insert – Updates to key fields will cause the original record in the Connection to be deleted and a new record with a new key added.<br>• Ignore – Do not update key fields in the Connection. If a key field is edited, the change will be stored with the Notes document but the key field(s) in the Connection will not change. |

## Intercept Document Delete Options

The Intercept Document Delete Options section of the RealTime Notes Activity document is shown below. The Intercept Document Delete Options section handles document deletions and removes the corresponding record from the external Connection.



| *Intercept Document Delete Field* | *Description* |
|---|---|
| Pre-Delete Formula | Notes formula language statement to execute on the Notes document immediately prior to the delete in the back-end database. |
| Activity(s) to Execute | Activity(s) to execute as part of the delete event. These are LEI Activities defined in the Administrator, and are executed immediately, regardless of Activity schedule settings. |
| Stored Procedure | Stored Procedure to execute in place of normal delete operation (input parameters are key values). |

## RealTime Notes Activity Considerations

Some important issues in using the RealTime Notes Activity are discussed below.

### Making Modifications to an Existing RealTime Notes Activity

When changing parameters on an existing RealTime Notes Activity:

1. Stop the RealTime Notes Activity process on the LEI server by either closing it from the Administrator or using the List command at the server console to determine the Activity's process ID number, and then using the Close [ID#] command to stop the Activity process. Do not use the Kill command.
2. Disable the schedule in the RealTime Notes Activity document.
3. Make the desired changes to the RealTime Notes Activity.
4. Re-enable the schedule and start the Activity from the Administrator.

### Stopping a Currently Running RealTime Notes Activity

Do not use the LEI server console Kill command to stop a RealTime Notes Activity. You must use the Close command. You can also use the Stop Activity hotspot on the Navigator to issue the Close command.

### Enabling RealTime

In order for RealTime Notes Activities to run properly, the RealTime Extension Manager library must be correctly set up and registered with your Domino server. Complete the steps below to set up the RealTime Extension Manager library.

**Note** If you selected the RealTime option during LEI installation, these steps should not be necessary.

1. In the notes.ini file for your Domino server, add the following line:
   ```
   EXTMGR_ADDINS=xxxEXT.DLL
   ```
   **Note** The name of xxxext.dll varies by platform as follows:
   - Windows NT: nleiext.dll
   - AIX: libleiext.a
   - Solaris: libleiext.so

   If an EXTMGR_ADDINS variable already exists in your notes.ini, add a comma and xxxext.dll to the existing entries, as follows:
   ```
   EXTMGR_ADDINS=SOME_FILE.DLL,ANOTHER.DLL,xxxEXT.DLL
   ```
2. The Domino server must be able to locate the LEIEXT library that you added to your notes.ini file in step 1 above. Make sure that the LEI program directory is in your path.
3. Run the LEI test program *rtntest.exe to verify your setup. See "Running rtntest.exe," below for more information.

**Note** When multiple notes.ini files are used, RealTime Notes requires that the Domino server use the same notes.ini file as LEI. Since LEI uses the default notes.ini, the Domino server must also.

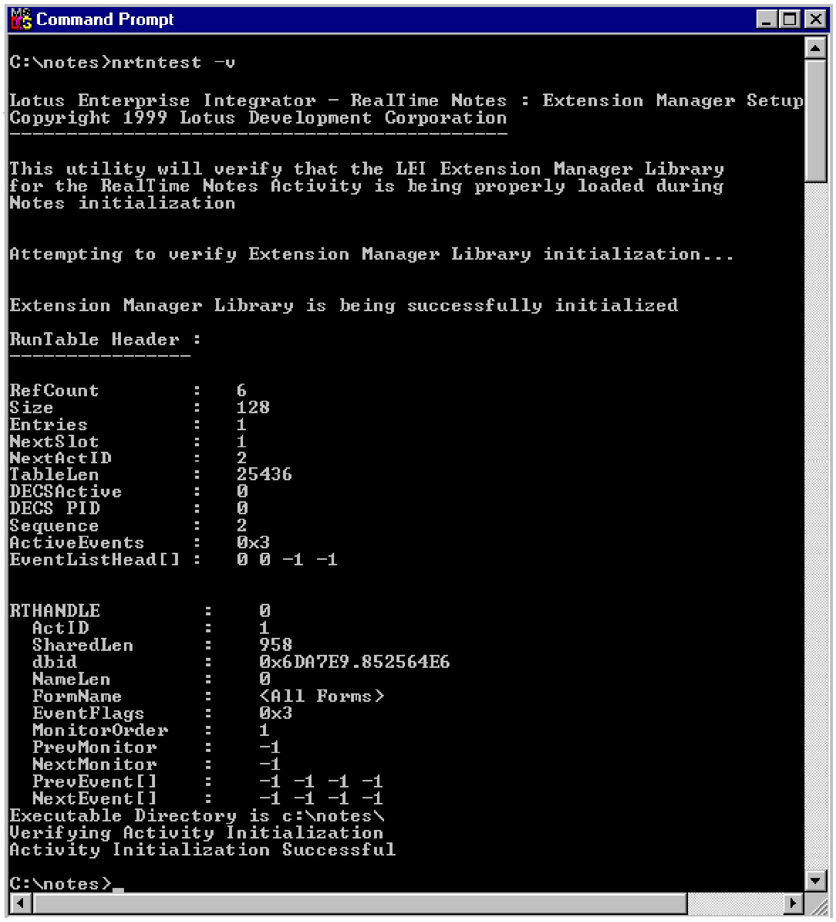
**Note** Although the Activity document appears to support it, a Notes to Notes RealTime Activity is not supported.

## Running rtntest.exe

The utility rtntest.exe verifies that you have successfully loaded the extension manager during Notes initialization. As with other connectivity tests, you enter *rtntest at the command prompt (where * is either n for Windows NT or rtntest without a prefix for UNIX). A message should appear, stating that the extension manager has been successfully installed.

There is an optional parameter, -v. If you add the -v parameter, you get more information, such as what forms are being monitored, what the monitor order of the forms is, and so on. It has two levels of information, depending on whether or not you have an active RealTime Notes Activity. If you do not have a RealTime Notes Activity running, you get some information but not the Activity-specific information present if an actual RealTime Notes Activity is running. If you have selected to monitor all forms (an option in the RealTime document under "Common Options"), you will see "<all forms>" as the FormName value, otherwise, it displays the form name or names.

Below is a screen shot of invoking nrtntest -v while a RealTime Notes Activity is running.


Command Prompt

```
C:\notes>nrtntest -v

Lotus Enterprise Integrator - RealTime Notes : Extension Manager Setup
Copyright 1999 Lotus Development Corporation
--------------------------------------------

This utility will verify that the LEI Extension Manager Library
for the RealTime Notes Activity is being properly loaded during
Notes initialization


Attempting to verify Extension Manager Library initialization...


Extension Manager Library is being successfully initialized

RunTable Header :
----------------

RefCount         :   6
Size             :   128
Entries          :   1
NextSlot         :   1
NextActID        :   2
TableLen         :   25436
DECSActive       :   0
DECS PID         :   0
Sequence         :   2
ActiveEvents     :   0x3
EventListHead[] :   0 0 -1 -1


RTHANDLE          :    0
  ActID           :    1
  SharedLen       :    958
  dbid            :    0x6DA7E9.852564E6
  NameLen         :    0
  FormName        :    <All Forms>
  EventFlags      :    0x3
  MonitorOrder    :    1
  PrevMonitor     :    -1
  NextMonitor     :    -1
  PrevEvent[]     :    -1 -1 -1 -1
  NextEvent[]     :    -1 -1 -1 -1
Executable Directory is c:\notes\
Verifying Activity Initialization
Activity Initialization Successful

C:\notes>_
```

## ODBC Data Sources and Drivers

When accessing an ODBC data source with a RealTime Notes Activity, you must use an ODBC driver and database client that support multithreading. The ODBC specification requires drivers to be thread-safe. However, in many instances drivers do not adhere to the standard.

**Caution** There are cases where the ODBC driver is thread-safe, but the database client is not thread-safe on a particular platform, for example, Oracle 7 on AIX.

When you set up a RealTime Notes Activity using an ODBC driver, you should first test run the Activity with the option Max. # of Concurrent Connections set to 1, and then try increasing it in subsequent tests. If increasing it causes crashes, you have a threading problem with your driver and should consult your ODBC driver provider.

## LotusScript Development and RealTime Notes Activity Considerations

The LEI RealTime functionality is installed in Domino as an extension manager library, which is always loaded by Domino. This module allocates an LEI Activity context in the Domino process – one that is a "nameless Activity." This supports greater performance and independence from the other LEI processes. All Notes processes initialize the extension manager, which initializes this process (this includes the Notes client).

When a scripted agent is run from within Notes client (for example, during script development on the LEI server), it is run in the same process as the RealTime Notes Activity module. It tries to create a "named Activity." This is incompatible with the already created "nameless Activity," and therefore fails. Suggestions for avoiding this are listed below:

1. Disable RealTime in the Notes client during script development. This means removing the extension manager entry from the notes.ini file and restarting the Notes client.
2. Make scripted Activities nameless (this provides no access to Connectors in the control store, and also no logging).
3. Run scripts as LEI scripted Activities instead. These are in a separate process, so are safe (however, debugging is not available).
4. Develop scripts on a machine other than the LEI server.

## Using RealTime Notes Activity on UNIX

When running LEI RealTime Notes Activities on UNIX, LEI must run under the same UNIX User ID as the Domino server. If you have an existing LEI installation that has a different UNIX User ID owning the LEI files from Domino, follow the steps below.

**Note** These steps assume the Domino server has the UNIX User ID of "notes", that the lei.ini file is in the directory /opt/lotus/lei, and that you did not select to install RealTime at Setup (when you installed LEI).

1. Log in as root and change the ownership of the LEI files to the Domino owner's ID.

   ```
   chown -R –notes /opt/lotus/lei
   ```

2. Modify the Notes shell initialization file to incorporate the required Shared Library Path and other environment settings required by LEI, as follows:

   - Locate your shell's initialization file:

     KSH: .profile and .kshrc (AIX users must modify LIBPATH in the .kshrc)

     CSH: .cshrc

     SH: .profile

   - Set the following environment variables:

     LANG=C

     Notes_ISOLATION=true

     LOTUS=/opt/lotus

     Notes_ExecDirectory=/opt/lotus/notes/latest/OSID (where OSID is "sunspa" for Solaris or "ibmpow" for AIX)

   - Set the PATH to include the following directories:

     Notes Resource directory:
     $LOTUS/notes/latest/OSID/res/$LANG

     Lotus executable directory: $Notes_ExecDirectory

     LEI directory: (for example, /opt/lotus/lei)

     Notes data directory: (so your notes.ini file can be found: for example,
     /home/user/notes).

- Set the Shared Library Path (AIX=LIBPATH, SOLARIS=LD_LIBRARY_PATH) to include the following directories:

  LEI directory: (for example, /opt/lotus/lei)

  Notes executable directory: $Notes_ExecDirectory

  Incorporate any other settings that are required by the external data sources that LEI will be loading. For instance, Oracle requires that ORACLE_HOME be set to the root of the Oracle installation and you must include the directory $ORACLE_HOME/lib in the Shared Library Path.

3. Modify the file notes.ini to include the statement

   ```
   EXTMGR_ADDINS=libleiext.EXT
   ```

   where **EXT** is the appropriate extension for your operating system (AIX=a, SOLARIS=so).
4. As the UNIX user "notes," execute the RealTime test program RTNTEST. This will verify that your environment is set up correctly for use with the RealTime Notes Activity. If you encounter problems, review the instructions above to make sure that you have properly configured your environment for use with the RealTime Notes Activity.
5. Verify your external database Connections.
   - Start a UNIX logon session as the Notes user to pick up any changes made to the shell initialization file.
   - Run the LCTEST program from the LEI directory to test connectivity to Sybase, ODBC, as so on.
   - If the LCTEST fails, verify that the native clients for that data source work. For example, with DB2, use the DB2 Command Window; with Oracle use SQLPlus, with Sybase use ISQL, and so on.

### Additional Instructions for Running RealTime on Solaris

On Solaris, the Domino server is installed by default with root privileges (SETUID root). This is done to allow Domino to raise the limit of file descriptors for the server process so that all server tasks can run as threads in a single process. After this limit is changed, the server no longer runs as root. However, as a side effect of the server process being configured as SETUID, the LD_LIBRARY_PATH is set to NULL. Because of this, the LEI Extension Manager shared library cannot be loaded, nor can the required shared libraries for any LEI external database Connections. Until this issue is resolved, you will need to remove the SETUID privileges for the Domino server. Depending on your server load, this may cause additional server processes to be started as additional Notes Connections are made. This may slightly impact performance of the Domino server. Following are instructions for removing and adding the SETUID privileges.

### How to Remove SETUID

As root, execute the command:

```
chmod u-s /opt/lotus/notes/latest/sunspa/server
```

### How to Add SETUID

As root, execute the command:

```
chmod u+s /opt/lotus/notes/latest/sunspa/server
```

## Monitoring More Than One Notes Form

There are several ways to monitor multiple Notes forms in the same database from one RealTime Notes Activity. Using the “Monitor Any Forms” Activity option will perform the actions on all forms that contain the same key fields. Another method involves creating a subform that contains the relevant data. Specifying that subform name in the Activity will allow Connection to the data from any form which uses that subform.

## Performance

High performance is crucial for the RealTime Notes Activity, since it may be serving many clients. The RealTime Notes Activity and the Extension Manager library invoked by the Domino server are optimized for performance. To maximize performance, when the external Connector is an SQL database, make sure an SQL index exists on the key fields. For example, if the key fields for a RealTime Notes Activity monitoring table “MyTable” are “FirstName” and “LastName”, create an index in the SQL database as follows:

```
CREATE UNIQUE INDEX IndexName on MyTable (FirstName, LastName)
```

When the LEI RealTime Notes Activity is enabled, LEI opens the Connection when the first event for that Activity is processed. LEI closes the Connection as soon as possible after the Activity has been scheduled to stop. By default, a single persistent Connection is shared by all users of the Notes database. It is possible to set the maximum number of concurrent, opened Connections to increase performance (see Max. Concurrent Connections options in the RealTime Notes Options section).

## RealTime Notes Activity Scheduling

RealTime Notes Activities can be run indefinitely by using the Start Activity or Run ASAP action button from the LEI Administrator Actions menu. If initiated using those methods, the Activities can only be closed from the Administrator process by using the Close command. To have an Activity run only during specific hours of the day, use the scheduling facility for the Activity to specify times to start and stop.

Unlike other Activities that continue their current run past any defined Stop Time until they complete, a RealTime Notes Activity, once started, runs until Stop Time and ceases.

When a RealTime Notes Activity is run via the Administrator, as a scheduled Activity, it recomputes its next invalid run time according to the scheduled stop time. However, when a RealTime Notes Activity is run otherwise, for example, using the Run ASAP Agent, the stop time is not recalculated. When a RealTime Notes Activity is run through a CGI or LEI server console command line program, it is not rescheduled and the invalid (stop) value is not updated.

If changes to the external database are occurring directly or through another interface besides the Notes front-ended RealTime Notes Activity, it will be necessary to occasionally stop the RealTime Notes Activity to run a Replication Activity to synchronize the key fields.

## Updating RealTime Fields in the Notes Document

Considerations for updating RealTime fields in the Notes Connector document are presented below.

- When updating data from Notes, only the key field data is required to be maintained in the Notes document. If the RealTime Data Storage options are used, all, or selected fields, can be saved to the Notes document when the document is closed. (See the Data Storage options in the RealTime Notes Options section.)
- Data in RealTime fields updated when the Activity is not running will be maintained in the Notes document until the next time the document is edited with the RealTime Notes Activity running, at which time they will be overwritten/removed (assuming the Update event is handled by the Activity).

- Block update does not allow updates to key fields in the Notes document or the Connection records. Delete/insert update causes the original record in the Connection to be deleted and a new record with a new key added. Ignore update does not update key fields in the Connection. If a key field is edited, the change is stored with the Notes document but the key field(s) in the Connection does not change.

## Stored Procedures

Stored procedures permit creation and execution of complex query actions against the Connection source, such as multi-table fetches and updates, and event handling. Use the Stored Procedures sections of the Event to name the stored procedure to be run when the event occurs.

A different stored procedure must be supplied for each event. The stored procedure for an event is executed when the event occurs, with an expected set of inputs and outputs, depending on the defined event selected. Any input parameters or output columns must be named based on the field names supplied in the Activity (for example, the field “MyField” supplies its value to a stored procedure parameter named “MyField”).

## Other Considerations When Configuring RealTime Notes Activities

The RealTime monitoring library does not enforce Notes access rules when processing a Delete event. As a result, the back-end Connector source record accessed via the RealTime Notes Activity-defined Notes document can be deleted, although the Notes database ACL might reject a deletion to the Notes form itself because the user does not have Author access (or higher) to the document.

If change or insert does not show up on the Notes side, check the LEI server Log file report for that Activity. Something has caused the insert/update to not occur.

When you use the Post-Open formula, actions occur, but because the Notes form doesn’t refresh automatically, you won’t see the changes until you close and reopen the document.

The following @ commands cannot be used in any Notes formula specified in the options of a RealTime Notes Activity (they are not supported except through the Notes interface): @DbLookup, @DbColumn, @Command, @MailSend, @Prompt, @DDEInitiate, @DDETerminate, @DDEExecute, @DDEPoke).

The Notes Connection Options field View to Use for Selection lets you specify a view used to select the documents that the Activity will use. When used with ordering in RealTime Notes Activities ("ordering" means that the Notes view must be ordered by the RT key fields), this field indicates the name of the view to use for the operation. Supplying a view name can drastically improve performance by not requiring that the database re-indexed every time the Activity is run. This feature only has an effect if Notes is the external database.

A RealTime Notes Activity that is run as the result of an event is always run immediately and locally. It is run as if it were entered from the command line, so command line restrictions apply (stop time is not computed).

When running a RealTime Notes Activities on parallel Domino clusters with separate LEI servers to the same back end, setup for the two Activities has to be identical. That is, field names must be in the same order.

## RealTime Dynamic Queries

The RealTime Notes Activity enables a Domino database to provide direct access to back-end data sources. This requires that one or more key field values be held in common with the Notes form (accessed by the Notes client or Web browser client) and the back end source. The key field values in the Notes form are used to query the back-end source data.

There are, however, many situations where the key field values will not already be available within the Notes database. In such cases, clients may want to freely insert the key values and then have the application retrieve the back-end data.

RealTime can still be used to retrieve information from the backend, but the user will have to know the key values in advance.

You can do this by creating a document in Notes using a RealTime-monitored form. After entering the Key value or values, save and then reopen the document in order to trigger a RealTime open event. You can do this programmatically in order to mask the save and reopen tasks from your user.

For more information, see the examples in Appendix E. In one example, a customer with a package tracking number opens a Notes form using a Web browser, enters, then clicks the Locate button. The RealTime Notes Activity then sends the tracking number to the back end database, which locates the package information using the tracking number. The matching record results are sent back to the client Web browser, which receives the status of the package.

# Chapter 26
# Replication Activity

This chapter provides information about the LEI Replication Activity. Included is information about the Activity, when to use it, and how to create the Activity. The chapter also describes how to resolve conflicts, which in LEI terms are simply instances where a data item in one database does not exactly match a data item in another database.

## Introduction to the Replication Activity

A Replication Activity synchronizes databases by updating one database from the data in another.

Replication involves metadata from two Connections (replication is done at the metadata level). There are two types of replication available within the Replication Activity form:

- Primary Key Replication
- Primary Key/Timestamp Replication

A condition can also be used in the Replication Activity document's Conditional Clause field to further refine the result set to achieve an enhanced, selective replication. See the section "Types of Replication Activities" later in this chapter for more information about these two types of replication.

## When to Use the Replication Activity

The Replication Activity is recommended for use when you want to:

- Synchronize data in two databases
- Merge data from one data source into another data source

You can also use the Replication Activity in conjunction with a Polling Activity to refine when the Replication Activity occurs. For example, you could poll a data source for a specific event, such as a data insertion, and on detection of that event start the Replication Activity that replicates the new data in another data source.

## Types of Replication Activities

There are several types of replication Activities.

### Primary Key Replication

LEI primary key replication replicates data based on a unique key common to both Connection sources that can be composed of one or more fields in the metadata. The function of the primary key is to determine if matching records exist in both data sources and if an update to a record, the insertion of a new record, or the deletion of a record is required.

The database identified in the Activity document as the master contains the data that “wins” any replication conflict.

When replication takes place, records are updated according to the following rules:

- If the keys match and the record is identical in both databases, no update is made to the record.
- If the keys match but the record doesn’t match, the record in the master database updates the one in the non-master database.
- If the primary key value exists in the master database but not in the non-master database, the record from the master database is inserted in the other database.
- If the primary key value exists in the non-master database but not in the master, the record from the non-master database is deleted.

Enabling the option Do Not Replicate Deletions in the Replication Activity document can prevent deletion of records from the non-master database.

### Keyed Replication

| *Link A* | *Link B* | *Master* | *Action* |
|---|---|---|---|
| data exists | doesn’t exist | A | insert into B* |
| | | B | |
| doesn’t exist | data exists | A | remove into A* |
| | | B | insert into A* |
| data exists | data exists | A | update B* |
| | | B | update A* |

*conflicts

## Primary Key/Timestamp Replication

If the metadata involved in the replication contains timestamp fields and these fields are identified in the Activity document, LEI performs Primary Key/Timestamp replication. For this purpose the LEI maintains a timestamp field that records the last time that a replication was begun. In this case the following process is used to replicate:

First the LEI creates two result sets (one for each database) that include only those records that have changed since the last replication based on the timestamp field. LEI then matches primary keys:

- If matching master and non-master records are found, the master record updates the other.
- If a record is found in either the master or non-master and there is no matching record in the other set, LEI attempts to locate the primary key in the other database. If the other key is found, the changed record updates the one that was not changed. If the other key is not found, a copy of the changed record is inserted into the other database.

Because LEI looks only at changed records, deletions are not replicated if a record is deleted from one database and the matching record in the other database remains unchanged.

The first time a Primary Key/Timestamp replication occurs, it behaves as a Primary Key replication since no replication timestamp has been set.

## Timestamp Replication

| *Link A* | *Link B* | *Master* | *Action* |
|---|---|---|---|
| new data | doesn’t exist | A | insert into B |
| | | B | insert into B |
| doesn’t exist | new data | A | insert into A |
| | | B | insert into A |
| change data | data exists | A | update B |
| | | B | update B |
| change data | change data | A | update into B* |
| | | B | update into A* |
| data exists | doesn’t exist | A | insert into B |
| | | B | insert into B |
| doesn’t exist | change | A | insert into A |
| | | B | insert into A |

*conflicts

## Using Timestamp Replication With Notes Databases

During Timestamp replication, LEI checks the time on the Domino Server(s) where the data source and destination reside. Note that the time that comes from the Domino Server may not be the same as the actual machine time where the Domino Server is running. It therefore becomes important to ensure that you use a timestamp field that captures the time on the Domino Server. Using @Modified retrieves the modified date/time for that document which was set by its Domino Server when the document was saved.

@Modified comes from the server. @Now comes from the desktop. Because the Domino Server time and the Notes Client workstation may not be in exact agreement about what time it is, it is probable that you will miss some documents during timestamp replication if the time on the client is earlier than the time on the server. Therefore you must use @Modified to obtain the proper time in any field formulas which will be used for timestamps. However @Modified is always one edit behind because the modified timestamp is the last thing to be set when the document is saved. @Modified reads the modification time, but because the document contains the last time the document was saved at the time the formula is executed, it will not return the time from the current save.

There are two options to solve this problem. The first is to use the Notes Connection Option “Load last modified TimeStamp (@Modified)”. In this case, @Modified would be entered into the timestamp field in the Replication Activity Form. This method requires no special changes to the Notes application. This option is discussed in more detail in the chapter on Notes Connections.

The second option involves modifying your Notes form to add a QueryClose event by performing the following steps:

In your timestamp field (which should be a computed field), use the following formula (where “EditedDate” is the fieldname of the field used for timestamp):

```
@IF(@IsDocBeingSaved;@Modified; EditedDate)
```

This formula places the Modification timestamp from the Domino Server into the field “EditedDate” if the document is saved, otherwise it doesn’t change.

Then to have the field contain the real Modified date, not the one from the previous edit, create a QueryClose event script for the form. This event will perform two saves of the document. The first will set the EditedDate field to the modified time of the last edit. The second save will set the modified time to the time of the save just performed. It is possible to get to within a 10th of a second of the real time.

```
Sub Queryclose(Source As Notesuidocument, Continue As
Variant)
     Dim workspace As New NotesUIWorkspace
     Dim doc As NotesUIDocument
     Set doc = workspace.currentdocument
     If doc.editmode Then
             Call doc.save
             Call doc.save
    Else
          Stop
    End If
End Sub
```

## Use of Database Triggers with Timestamp Replication

Database applications often use triggers to set timestamps when a row of data has been modified in some way. When LEI replication updates a row, that may also cause the trigger to reset the timestamp to the time modified in that file or database. This may trigger unnecessary replications of this row back to the other database the next time replication is run. The trigger should be designed to detect if it is the client application or the LEI updating the row and deal with the timestamp accordingly.

## Importance of Sort Order

LEI sorts the data it accesses in order to match the primary keys. If the sort order used in two databases are not the same, mismatches can cause unnecessary insertions and deletions. Make certain that the sort order specified for the databases used in the replication are identical. One option for handling sort order differences in replication is to use a numeric key. Using Order MetaConnections for databases which sort differently will produce results without degradation of performance. See the Order MetaConnector information in Chapter 16 for more information about their uses. For issues arising from sorting differences based on case, see the section on Replication Options below on case-insensitive string comparison.

## Specifying a Selection View in a Notes Database

Timestamp replication creates a temporary view based on the selection criteria of documents to be replicated. The Notes Connection Options field "View to Use for Selection" lets you specify a permanent view to be used to select the documents that the Activity will use. Supplying a permanent view name can dramatically improve performance by not requiring the database to be re-indexed every time the Activity runs. LEI requires the ability to modify this view, so it should not be a view that users also access.

## Sort Order Considerations

Replication produces ordered result sets from both Connections, and compares those result sets. If the Connections use different sort orders, then the comparisons may be inaccurate, producing unnecessary insert/delete pairs. The Order MetaConnector is available to impose a consistent sort order on result sets. Since there is a performance cost to using this MetaConnector, it should only be used when the following conditions are met, in which case it must be used for both Connections:

- The data being replicated is not on the same system. If both Connections are to the same Connector (for example, DB2 to DB2 replication), the Order MetaConnector is not needed.
- One or more keys are text. Sort order is not a problem for non-textual keys.
- The Connections in the replication sort differently. This varies by Connector. Some may use the same sort order even though they connect to different systems.
- The text keys contain characters that are included in those which sort differently. For example, Lotus Notes and Oracle may sort certain punctuation characters differently, but if the text keys are purely alphanumeric, the sort order differences may be irrelevant for this replication.

**Note** When data is obtained from an EBCDIC-based system, the sort order will almost always differ from a non-EBCDIC system.

## How to Create a Replication Activity

The Replication Activity document defines information needed for a Replication Activity. This information includes:

- Replication parameters
- Options controlling the treatment of the data

The steps below outline the general procedure for creating a Replication Activity. Refer to the other sections in this chapter for more information about the particular fields and options in the Replication Activity document.

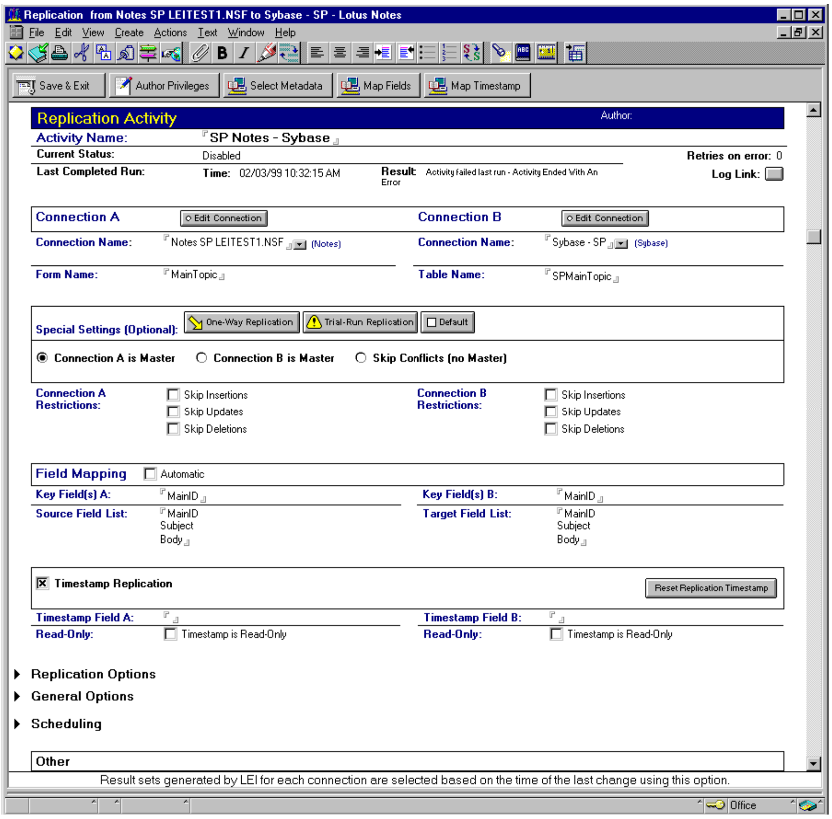
1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – Replication from the menu bar, or click Create Activity on the navigator and then select Replication.
3. Fill in the required fields and select the desired options for the Activity in the Replication Activity document, including its Connections and Scheduling options. All Connections that you want to use in the Activity must already exist and should have been tested using the LCTEST described in Chapter 5, "Introduction to Connectors".
4. Save the Activity document.

You can select the Run ASAP option to have the Activity executed by the LEI Server as soon as possible after saving the Activity.

**Note** There is a complete example of building a Replication Activity in Appendix E, Tutorials.

## Replication Activity Document

Choose Create – Activity – Replication from the menu bar, or click Create Activity on the navigator and select Replication to access the Replication Activity document, shown below. The fields and options in the document are described below.


Replication from Notes SP LEITEST1.NSF to Sybase - SP - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields
Map Timestamp
Edit Connection
One-Way Replication
Trial-Run Replication
Default
Reset Replication Timestamp
Office

Replication Activity — Author:

Activity Name: SP Notes - Sybase
Current Status: Disabled — Retries on error: 0
Last Completed Run: Time: 02/03/99 10:32:15 AM — Result: Activity failed last run - Activity Ended With An Error — Log Link:

| Connection A | | Connection B | |
|---|---|---|---|
| Connection Name: | Notes SP LEITEST1.NSF (Notes) | Connection Name: | Sybase - SP (Sybase) |
| Form Name: | MainTopic | Table Name: | SPMainTopic |

Special Settings (Optional):

[✓ Connection A is Master] Connection B is Master Skip Conflicts (no Master)

| | | | |
|---|---|---|---|
| Connection A Restrictions: | ☐ Skip Insertions<br>☐ Skip Updates<br>☐ Skip Deletions | Connection B Restrictions: | ☐ Skip Insertions<br>☐ Skip Updates<br>☐ Skip Deletions |

Field Mapping ☐ Automatic

| | | | |
|---|---|---|---|
| Key Field(s) A: | MainID | Key Field(s) B: | MainID |
| Source Field List: | MainID<br>Subject<br>Body | Target Field List: | MainID<br>Subject<br>Body |

☒ Timestamp Replication

| | | | |
|---|---|---|---|
| Timestamp Field A: | | Timestamp Field B: | |
| Read-Only: | ☐ Timestamp is Read-Only | Read-Only: | ☐ Timestamp is Read-Only |

- Replication Options
- General Options
- Scheduling

Other

Result sets generated by LEI for each connection are selected based on the time of the last change using this option.

See Chapter 17, “Introduction to LEI Activities”, for descriptions of the following sections, which are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

## Connections A and B

The Connectors section provides information on the data source and destination. Connectors must be created before they can be used in an Activity document.

| *Field* | *Description* |
|---|---|
| Edit Connection | The Edit Connection button (located in the blue title bar directly over the selected Connection's name) brings up the selected Connection's document in Edit mode. This button is available only if you are in Edit mode and after you have entered a Connection name. |
| Connection Name | Names of the two Connections. The order of the Connections (which is A and which is B) does not matter. The Connection identifies the server and database with access information. The Additional Information segment of the document (see below) identifies the metadata.<br>To display a list of available Connections, click on the arrow to the right of the edit field. Alternatively, you can place the cursor in the edit field and press Enter to see the list of Connections. You can also place the cursor in the edit field and press a letter key to see the first Connection beginning with that letter. Typing additional letters will move the selection to match the typed letters.<br>After selecting the Connections, you can browse the associated databases for metadata and field information by clicking the "Select Metadata" and "Map Fields" Action buttons. |
| Form/Table Name | The metadata, based on the type of Connection chosen. For example, in a relational database the metadata is a Table, in a Notes database the metadata is a Form. |

## Special Settings

The following settings are optional.

| *Button* | *Description* |
|---|---|
| One-Way Replication | If you click this button, LEI will automatically format part of the Replication Activity document for you so that data transfers from one database to the other only. All "Connection Restrictions" options are marked for the Connector you select as the master. |
| Trial-Run Replication | If you click this button, you can run and log the replication Activity without making any changes to the databases involved. Also, all conflicts are logged. |

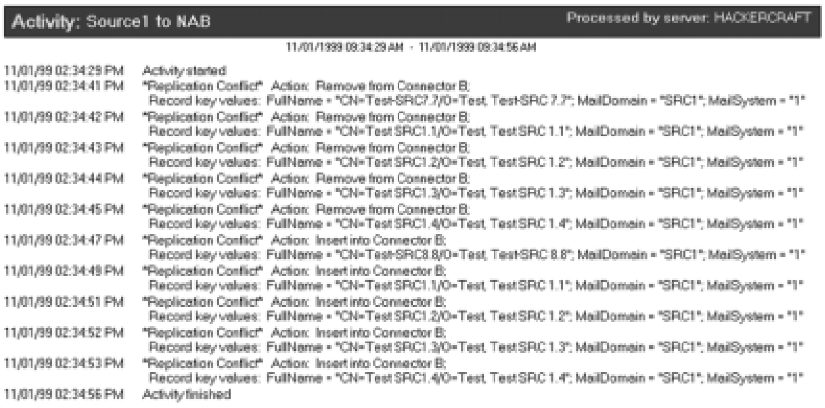
*continued*

| *Button* | *Description* |
|---|---|
| Default | Standard replication clears the fields associated with One-Way and Trial-Run Replication, effectively returning some sections of the Replication Activity document to their default settings. This deselects the Skip Conflicts option, deselects all Connection Restrictions, and sets Connection A as master. |

The Replication Activity document provides options for specifying how to resolve conflicts that occur while the Activity runs.

The term conflict, as it pertains to LEI, specifically refers to a data item in one database not exactly matching a data item in another database. A conflict occurs when the timestamp cannot resolve a "winner". In keyed replication, all differences are categorized as conflicts. In a timestamped Activity, only certain discrepancies are categorized as conflicts. Conflicts can be a normal event during replication. A decision must then be made as to how to resolve this conflict.

**Note** You can optionally choose to log replication conflicts to a log file. These logged entries are not errors, but are rather instances of a conflict event. Listing name conflicts to a log file helps you to both check the Activity and resolve any naming mismatches between the two databases. A sample conflict log excerpt is shown below.



```
Activity: Source1 to NAB                                        Processed by server: HACKERCRAFT
                          11/01/1999 09:34:29 AM - 11/01/1999 09:34:56 AM

11/01/99 02:34:29 PM   Activity started
11/01/99 02:34:41 PM   *Replication Conflict*  Action: Remove from Connector B;
                        Record key values: FullName = "CN=Test-SRC7.7/O=Test, Test-SRC 7.7"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:42 PM   *Replication Conflict*  Action: Remove from Connector B;
                        Record key values: FullName = "CN=Test SRC1.1/O=Test, Test SRC 1.1"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:43 PM   *Replication Conflict*  Action: Remove from Connector B;
                        Record key values: FullName = "CN=Test SRC1.2/O=Test, Test SRC 1.2"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:44 PM   *Replication Conflict*  Action: Remove from Connector B;
                        Record key values: FullName = "CN=Test SRC1.3/O=Test, Test SRC 1.3"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:45 PM   *Replication Conflict*  Action: Remove from Connector B;
                        Record key values: FullName = "CN=Test SRC1.4/O=Test, Test SRC 1.4"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:47 PM   *Replication Conflict*  Action: Insert into Connector B;
                        Record key values: FullName = "CN=Test-SRC8.8/O=Test, Test-SRC 8.8"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:49 PM   *Replication Conflict*  Action: Insert into Connector B;
                        Record key values: FullName = "CN=Test SRC1.1/O=Test, Test SRC 1.1"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:51 PM   *Replication Conflict*  Action: Insert into Connector B;
                        Record key values: FullName = "CN=Test SRC1.2/O=Test, Test SRC 1.2"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:52 PM   *Replication Conflict*  Action: Insert into Connector B;
                        Record key values: FullName = "CN=Test SRC1.3/O=Test, Test SRC 1.3"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:53 PM   *Replication Conflict*  Action: Insert into Connector B;
                        Record key values: FullName = "CN=Test SRC1.4/O=Test, Test SRC 1.4"; MailDomain = "SRC1"; MailSystem = "1"
11/01/99 02:34:56 PM   Activity finished
```

LEI has many options that allow you to control how replication conflict events are handled. Each of these options is described in the table below.

| *Option* | *Description* |
|---|---|
| Connection A is Master | (Default) Uses Connection A as the master. The master Connection is the one that wins any replication conflict. |
| Connection B is Master | Uses Connection B as the master. |
| Skip Conflicts (No Master) | Does not update either Connection on replication conflicts. |

## Connection Restrictions

The Restrictions section provides options that you can choose from to restrict the operational behavior of the Replication Activity relative to each Connector. For an extended example of how to work with restriction settings, see “Example Results for Restricted Settings” later in this chapter.

| *Option* | *Description* |
|---|---|
| Skip Insertions | Prevents any records in the associated database from receiving inserts during replication. |
| Skip Updates | Prevents any records in the associated database from being updated during replication. |
| Skip Deletions | Prevents any records in the associated database from being deleted during replication. |
|  | Enabling No Insertions, No Updates, and No Deletions causes replication for that Connection to NOT allow writeback modifications. This reduces the locking level, and may be used to avoid locking one side of a replication when it is known that no changes will be made to that side. It is especially useful when replicating from non-updateable views in a relational database. |

## Field Mapping Options

The Field Mapping section of the Replication Activity document is user defined, unless you click the Automatic check box. See the table below for a description of the options. Also see the Data Mapping section in Chapter 17, Introduction to LEI Activities".

| *Option* | *Description* |
|---|---|
| Key Fields | The fields or columns of the data that together represent a unique primary key for that data collection. No two records can have the same value for all key fields, and at least one key field is required. The key fields for both Connections must correspond. |
| Source/Target Field List | This option is only available if you do NOT click the Automatic check box. These lists must correspond on a one-to-one basis. That is the Source Field List must correspond with the Target Field List. You can either enter the field names into this field, or use the "Map Fields" Action button to select fields to list.<br>If a field in the source database does not have a corresponding field in the destination database, that field is ignored. Fields that exist in the destination database but have no corresponding field in the source database are also ignored. |
| Map Fields: By Field Name | This option is only available if you DO click the Automatic check box. Maps data from each source column to a identically named column in the destination metadata.<br>If a field in the source database does not have a corresponding field in the destination database, that field is ignored. Fields that exist in the destination database but have no corresponding field in the source database are also ignored. |
| Map Fields By Field Position | This option is only available if you DO click the Automatic check box. Maps data from each source column to the destination by column sequence: first column to first, second to second, and so on.<br>When the number of fields in the source and destination databases are different, excess fields are ignored. |

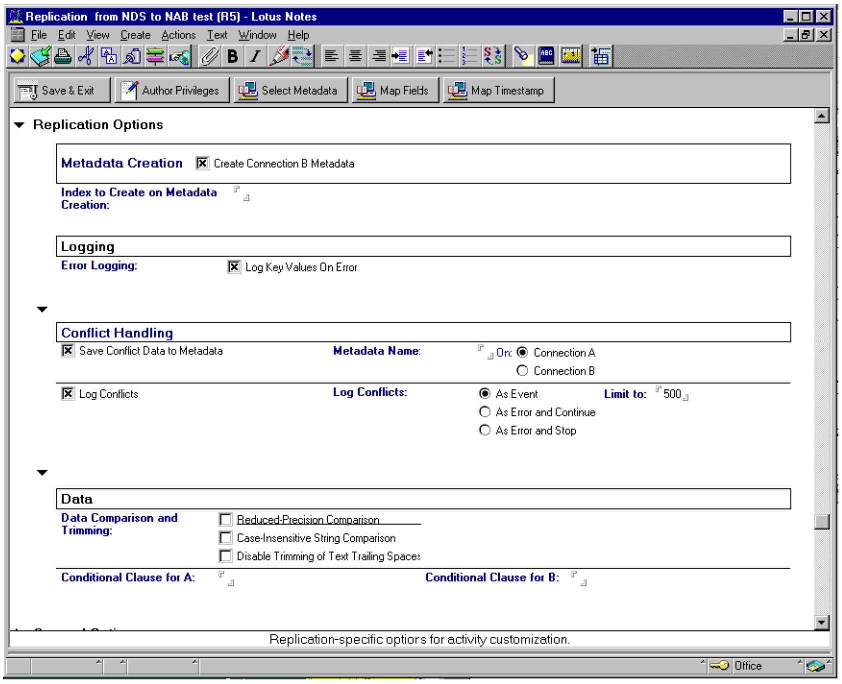
## Timestamp Replication

If you click the Timestamp Replication check box, the following fields become available to you.

| *Field* | *Required* | *Description* |
|---|---|---|
| Reset Replication Timestamp | No | Clicking this button reinitializes the replication timestamp of both metadata objects that identifies the last replication time. Effectively this forces the next Primary Key/Timestamp replication to behave as though no previous replication has occurred. The option is useful if replication has fallen out of sync. |
| Timestamp Field | Required to do Timestamp Replication | This specifies the name of the field that holds the record timestamp. If a field is entered here, LEI compares both the timestamp and the primary key. If used, a timestamp field name must be entered for both Connections. If Timestamps are omitted, LEI uses the primary key rules to replicate the database. |
| Timestamp is Read-only | Optional | This sets the timestamp field for the Connector as read-only for LEI. With this option selected, the rest of the data is checked for changes. It prevents records which are essentially unchanged (except for the timestamp) from being replicated over and over. LEI will not alter the field; any change is left to the data source to manage. |

## Replication Options

There are several options available to you in the Replication Options section of the Replication Activity document. They are all optional and they are broken down into subsections: Metadata Creation, Logging, Conflict Handling, and Data. Each subsection is described below.


Replication from NDS to NAB test (R5) - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields
Map Timestamp
Office

Replication Options

Metadata Creation [✓ Create Connection B Metadata]

Index to Create on Metadata Creation:

Logging

Error Logging: [✓ Log Key Values On Error]

Conflict Handling

[✓ Save Conflict Data to Metadata] Metadata Name: On: [✓ Connection A] Connection B

[✓ Log Conflicts] Log Conflicts: [✓ As Event] As Error and Continue As Error and Stop Limit to: 500

Data

Data Comparison and Trimming: Reduced-Precision Comparison; Case-Insensitive String Comparison; Disable Trimming of Text Trailing Spaces

Conditional Clause for A: Conditional Clause for B:

Replication-specific options for activity customization.

## Metadata Creation and Logging

| *Option* | *Description* |
|---|---|
| Create Connection B Metadata | LEI will create a new metadata object in the database identified in the non-master Connection to hold the data being transferred. In creating a new metadata, LEI will try to match the master metadata as much as possible. This option is only available if metadata will be created by the Activity. It creates an index for use by LEI to do data comparisons. |
| Index to create on Metadata Creation | When the option to create metadata is selected and metadata creation is performed, an index will also be created on the new metadata. The name entered is the name for the new index, and the replication Activity key fields are the index columns. Index creation may not be supported by all Connectors. An index is crucial for good performance against some Connectors, such as relational databases. |
| Log Key Values on Error | Log Key Values on Error: Creates an entry in the Activity Log when an error occurs during replication. The entry contains the keys from the record that generated the error. (This option assists in troubleshooting.) |

## Conflict Handling

| *Option* | *Description* |
|---|---|
| Save Conflict to Metadata | Saves a document containing the conflict in the non-master metadata. When selecting this option, also indicate:<br>Metadata Name – Enter the metadata to receive conflict loser records.<br>Connection on which the metadata resides. |
| Log Conflicts | Logs conflicts in the Activity log. Selecting Log Conflicts reveals these options:<br>As Event – Logs conflicts as errors in the Activity Log.<br>As Error and Continue – Logs conflicts as errors in the Activity Log, but continues the Activity.<br>As Error and Stop – logs conflicts as errors in the Activity Log and stops the Activity.<br>Logging conflicts with As error is slower, since each log entry is written to disk. As event is much faster and recommended for production, but a crash loses conflict information. |
| Limit to | The maximum number of conflicts to log is used to limit logged conflicts to a useful number for the given Activity. When this number of conflicts has been logged, additional conflicts are not added to the Activity log. The default is 500, and a value of zero indicates no limit (log all conflicts). |

## Data

| *Option* | *Description* |
|---|---|
| Reduced-Precision Comparison | Avoids false mismatches as a result of differences between database types in the precision used for storing datetime or timestamp and floating point values. One database, for example, may recognize fractions of seconds and another not. This option uses a lower common degree of precision. Datetimes are compared to the second and without timezone or daylight savings time. Does date/time comparisons based on year, month, day, hour, minute, and second only. Does floating-point comparisons to ten digits of precision. Does NOT do comparisons based on timezones, daylight savings time, or hundredths of a second. It is useful for replication between databases that do not support the same degree of date/time floating point precision. |
| Case-Insensitive String Comparison | This option is intended for use only when both Connections in a replication use case-insensitive sorting (such as LEI replication between two Notes Connections). In this situation, normal replication case-sensitive comparisons can cause false conflicts. Using this option in such cases causes replication to compare text without case sensitivity<br><br>**Note** The option does not cause the Connections to perform case-insensitive sorting. |
| Disable Trimming of Text Trailing Spaces | Text fields that contain trailing spaces will not be trimmed. Trimming is the default behavior. |
| Conditional Clause for A and B | Optional. Used to further refine the Activity by being included in the statement being executed against the result set based on the key field(s), such as, a SQL WHERE clause or Notes selection formula clause. A condition can be entered for either, or both, Connections. The syntax of a conditional clause is defined by the Connection, however, it is not necessary to include any keywords (such as SELECT or WHERE). |

## Example Results for Restriction Settings

The following table describes the results for various restriction settings.

| *Connection A Setting* | *Connection B Setting* | *Action* | *Result* |
|---|---|---|---|
| Timestamp is Read-only | | Change A timestamp only | Not updated in B |
| | | Change A timestamp and fields | Data only updated in B |
| | | Change B timestamp only | Update timestamp only in A |
| | | Change B timestamp and fields | Data and timestamp updated in A |
| | Timestamp is Read-only | Change A timestamp only | Update timestamp only in B |
| | | Change A timestamp and fields | Data and timestamp updated in B |
| | | Change B timestamp only | No updates in A |
| | | Change B timestamp and fields | Data only updated in A |

If the Connection is using a timestamp that is set by the backend database through a trigger or as a result of saving the document, LEI has no effect on that timestamp. It is set per the rules of the backend database. As an example, when using the "Modified" time on a Notes document, selecting the "Load Last Modified Timestamp (@Modified)" Connection Option will force LEI to use the Modification time on the Notes document. If the document is changed during a replication, Notes will update that modification time as the document is saved.

When a user or data entry application outside of LEI enters new data or changes record data in a database, a timestamp is updated to reflect the access date and time at which the latest modification occurred. For LEI, the timestamp field for the target database can either be read-only or can actually be updated by LEI. You can control whether or not LEI updates the timestamp field in the target database.

This example table shows results when both Connections are Notes and are using the "Load Last Modified Timestamp (@Modified)" Connector Option.

| *Connection A Setting* | *Connection B Setting* | *Action* | *Result* |
|---|---|---|---|
| Timestamp is Read-only | | Change A timestamp only | Timestamp untouched in B |
| | | Change A timestamp and fields | Data only updated in B, timestamp set to time of document change (B is now later than A) by Notes when document was modified. |
| | | Change B timestamp only | Updates timestamp only in A |
| | | Change B timestamp and fields | Data and timestamp updated in A |
| | Timestamp is Read-only | Change A timestamp only | Timestamp updated in B (B is now later than A) |
| | | Change A timestamp and fields | Timestamp and fields updated in B |
| | | Change B timestamp only | No updates to A |
| | | Change B timestamp and fields | Updated A fields and timestamp |
| Timestamp is Read-only | Timestamp is Read-only | Change B fields and timestamp | Update A fields and timestamp |
| | | Change A fields and Timestamp | Update B fields and timestamp |
| | | Change B timestamp only | No updates to A |
| | | Change A timestamp only | No updates to B |

## Using Restrictions in Keyed Replication Activities

Settings for replication restrictions apply to the Connection for which they are set. For example, when options are set on the destination Connection, changes to the source will not affect the destination for the specified action. Note that in keyed replication, setting any flags on the Master side has no effect as it produces the same result as a normal Keyed Replication.

The following tables represent the effect of setting various Replication Restrictions on each Connection. The action performed when each option is set will produce the specified results to the data in the Connection for which the option is set. The type of replication also determines behavior of the destination. Keyed replications behave differently than timestamp replications. In any case, all of these options are used to manipulate the data transfer in a way that will cause the data to be different on each side of the replication.

### Connection A is the Master

| *Connection B Setting* | *Action* | *Result* |
|---|---|---|
| Skip Insertions | Insert in A | No insert in B |
| | Insert in B | Remove from B |
| Skip Updates | Update in A | No update in B |
| | Update in B | Replaced with A |
| Skip Deletions | Delete from A | Retained in B |
| | Delete from B | Replaced with A |

### Connection B is the Master

| *Connection A Setting* | *Action* | *Result* |
|---|---|---|
| Skip Insertions | Insert in B | No insert in A |
| | Insert in A | Removed from A |
| Skip Updates | Update in B | No update to A |
| | Update in A | Replaced with B |
| Skip Deletions | Delete from B | Retained in A |
| | Delete from A | Replaced with B |

**Note** If you see random blocks of data deleted or inserted during a keyed replication Activity (especially one taking longer than 10 minutes), the problem may have to do with the update process in Domino 4.6.3 or 4.6.4. To solve the problem, turn the update task off in the Domino server while the Activity runs, or turn it off altogether by removing the task from the notes.ini file. To reinsert deleted data, simply rerun the replication Activity until 0 (zero) inserts and 0 (zero) removes are achieved.

## Using Restrictions in Timestamp Replication

Since Timestamp Replication is bi-directional (A updates B and B updates A), it is possible to set restrictions for either Connection. “Skip Deletions” is required on both A and B Connections for all Timestamp Replications, since deletions are never propagated to the other Connection. It is assumed in this chart that Skip Deletions is set and all deletions are ignored by each Connection.

### When Connection A is the Master

| *Connection A Setting* | *Action* | *Result* |
|---|---|---|
| Skip Insertions and Deletions | Insert in A | Inserts in B |
| | Update in A | Updates B |
| | Insert in B | No insert in A |
| | Update in B | Updates A |
| Skip Updates and Deletions | Insert in A | Insert in B |
| | Update in A | Updates B |
| | Insert in B | Insert in A |
| | Update in B | No update in A |
| Skip Inserts, Updates and Deletions | Insert in A | Inserts in B |
| | Update in A | Updates B |
| | Insert in B | No insert in A |
| | Update in B | No update in A |

| *Connection B Setting* | *Action* | *Result* |
|---|---|---|
| Skip Insertions and Deletions | Insert in A | No insert in B |
| | Update in A | Updates B |
| | Insert in B | Insert in A |
| | Update in B | Updates A |
| Skip Updates and Deletions | Insert in A | Inserts in B |
| | Update in A | No updates in B |
| | Insert in B | Inserts in A |
| | Update in B | Updates A |

*continued*

| *Connection B Setting* | *Action* | *Result* |
|---|---|---|
| Skip Inserts, Updates and Deletions | Insert in A | No insert in B |
| | Update in A | No update in B |
| | Insert in B | Inserts in A |
| | Update in B | Updates A |

## When Connection B is the Master

| *Connection A Setting* | *Action* | *Result* |
|---|---|---|
| Skip Inserts and Deletions | Insert in A | Inserts in B |
| | Update in A | Updates B |
| | Insert in B | No insert in A |
| | Update in B | Update in A |
| Skip Updates and Deletions | Insert in A | Inserts in B |
| | Update in A | Update in B |
| | Insert in B | Inserts in A |
| | Update in B | No update in B |
| Skip Inserts, Updates and Deletions | Insert in A | Inserts in B |
| | Update in A | Updates in B |
| | Insert in B | No insert in A |
| | Update in B | No updates in A |

| *Connection B Setting* | *Action* | *Result* |
|---|---|---|
| Skip Inserts and Deletions | Insert in A | No insert in B |
| | Update in A | Updates B |
| | Insert in B | Inserts in A |
| | Update in B | Updates A |
| Skip Updates and Deletions | Insert in A | Insert in B |
| | Update in A | No update in B |
| | Insert in B | Insert in A |
| | Update in B | Update in A |
| Skip Inserts, Updates and Deletions | Insert in A | No insert in B |
| | Update in A | No update in B |
| | Insert in B | Inserts in A |
| | Update in B | Updates A |

## Using Replication on AS/400

The sections below describe aspects of the Replication Activity which are specific to running LEI on the AS/400.

### Deletions

Provided that the connected-to databases are local to an AS/400 LEI server and that they are Notes or DB2/400 specific databases, support to capture deletes is optionally available. This support results in deleted documents or records being captured at delete time and moved to a staging area (table or form) to be read and applied to the corresponding replica database the next time the Activity is run.

DB2/400 deletes are captured in an LEI generated table in library QNOTE-SEIDT and Notes document deletes are captured in a Notes form (metadata) generated by LEI and stored in the same database as the source Notes database. These documents or rows are only captured locally as they are captured at the time of the delete request via a DB2/400 trigger program or Lotus Notes Extension manager program.

The first time you run with the Capture Deletes option checked, the capture delete metadata is created and the metadata name will be viewable from the Replication Activity form. You will also be given instructions in the Log output on how to set up the DB2/400 trigger program or the Notes Capture Delete Extension Manager.

You can remove the delete capture request by simply unchecking the delete capture checkbox and running the Replication Activity again. When the Activity is run, all captured data will be purged, as will the table and the form. Note that removing the capture delete option from a DB2/400 table will not automatically remove the DB trigger program from your table. You will need to execute the corresponding RMVPFTRG command.

**Note** The ADDPFTRG and RMVPFTRG requests are done manually because they can only be done at a time when the table is not “in use” and this is best determined by you. If you already have a delete trigger program associated with the DB2/400 table, you will not be able to add another. Only one delete after trigger is allowed.

To set up a DB2/400 database trigger to capture the deletes, do the following:

```
ADDPFTRG  (library_name/file_name)  TRGTIME(*AFTER)
TRGEVENT(*DELETE) PGM(QNOTESLEI/LEIDTRG)  ALWREPCHG(*YES)
```

To set up the capture delete extension manager for Notes, add the following (or to the following) in the notes.ini:

```
EXTMGR_ADDINS=..,...,leidext
```

Note that the log statistics for Replication with delete captures will reflect the fetch and the removal of the staged capture delete data.

### Conflict Resolution

Check one of the following options:

**Connection A is Master** Uses Connection A as the master. The master Connection is the one that wins any replication conflict.

**Connection B is Master** Uses Connection B as the master.

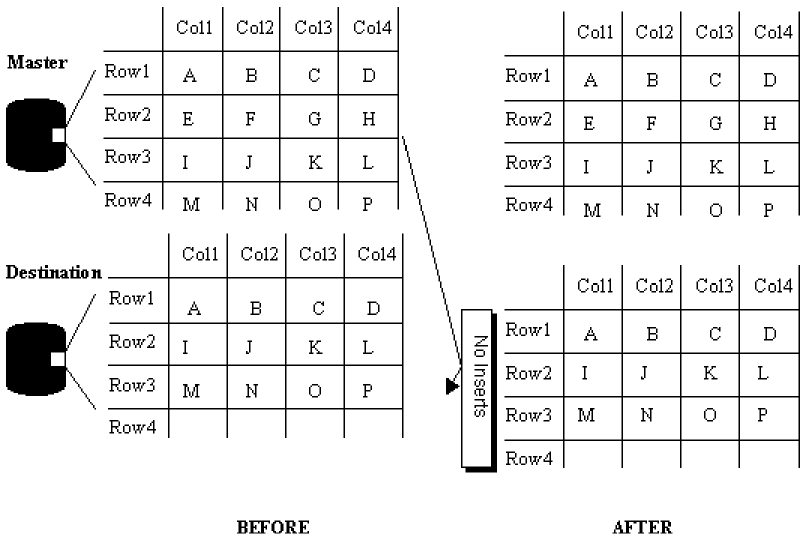
**Timestamp is Master (AS/400 only)** Uses either Connection A or Connection B as the master depending on the more recent timestamp in the document or row.

## Replication Examples

This section provides some examples that employ the Replication Activity with replication restrictions.

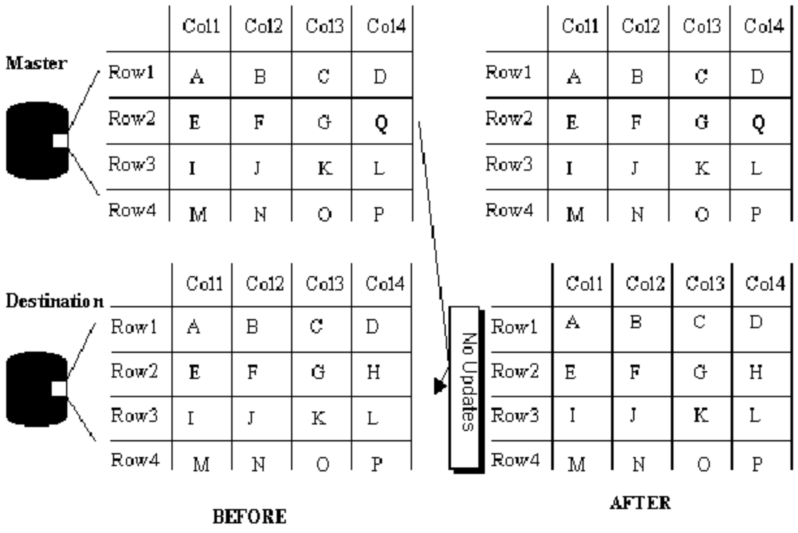
### Skip Insertions

In this example, "Skip Insertions" is set on the Connection B side and a record is inserted into Connection A. The insertion is ignored by Connection B in the next replication. Use this option to prevent new records from being inserted to a destination database.



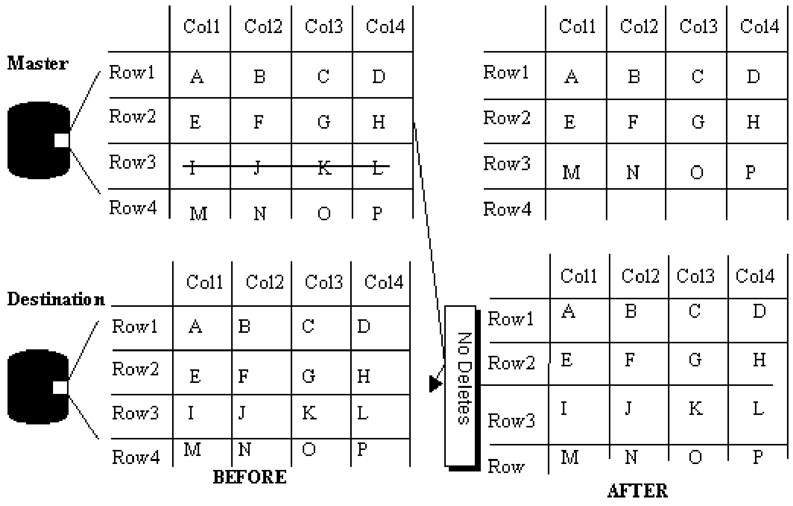
## Skip Updates

In this example, “Skip Updates” is set. A record is updated in the master but is not replicated to the destination. Use this option to retain original data.



## Skip Deletions

The example below shows the effect of “Skip Deletions.” The master has deleted a record but the deletion is skipped in the destination. Use this option to preserve data in the destination that has been deleted from the master.



# Chapter 27
# Java Activity

This chapter provides information about the LEI Java Activity. Included is information about this Activity, when to use it, and how to create it.

## Introduction to the Java Activity

A Java Activity allows LEI to launch a Java application.

## When to Use the Java Activity

Use the Java Activity any time you want to utilize LEI to launch a Java application. The Java application can be anything you want, and in fact, it need not be restricted to LEI-specific applications. Generally, however, the Java application utilizes the LEI Java classes shipped with LEI, which expose the entire LEI Application Programming Interface. If you are already familiar with the Scripted Activity using LotusScript and the LC LSX or LEI LSX, this is the same concept.

Creating a Java application using the LEI classes or using LotusScript with the LEI LotusScript Extensions both provide powerful programmatic control over LEI capabilities. Using Java, however, gives a programmer the power of a full-featured, robust and portable object oriented language in which to develop LEI applications. Further, the programmer is free to choose an IDE of his or her choice to develop and deploy applications.

There are two restrictions to keep in mind if choosing to develop an LEI Java Activity. First, the application must exist on the same machine as the LEI server that is designated to run the application. This means that you must designate a specific server in the Java Activity form. Second, a Java JVM (JDK 1.1 or JAVA 1.2 compatible) must also exist on that machine.

### Example Usage

Typical use for an LEI Java application would be a situation where you need more elaborate control of source and destination during data transfers, or performing different kinds of data massaging based on certain application specific conditions. This type of finely tuned control can only be performed using the LEI API via the LEI Java Classes (or the LEI LSX under LotusScript).

Java provides the richness of a full fledged programming language.

## How to Create a Java Activity

The steps below outline the general procedure for creating a Java Activity in LEI. Refer to the other sections in this chapter for more information about the particular fields and options in the Activity document.

1. Open the LEI Administrator database, leiadm.nsf.
2. Choose Create – Activity – Java from the menu bar, or click Create Activity in the navigator and select Java.
3. Fill in the required fields and select the desired options for the Activity in the Java Activity document, including its Scheduling options.
4. Save the Activity document.

You can select the Run ASAP option to have the Activity executed by the LEI Server as soon as possible after saving the Activity.

**Note** The class specified in the Class Name field should be in the CLASSPATH environment variable.

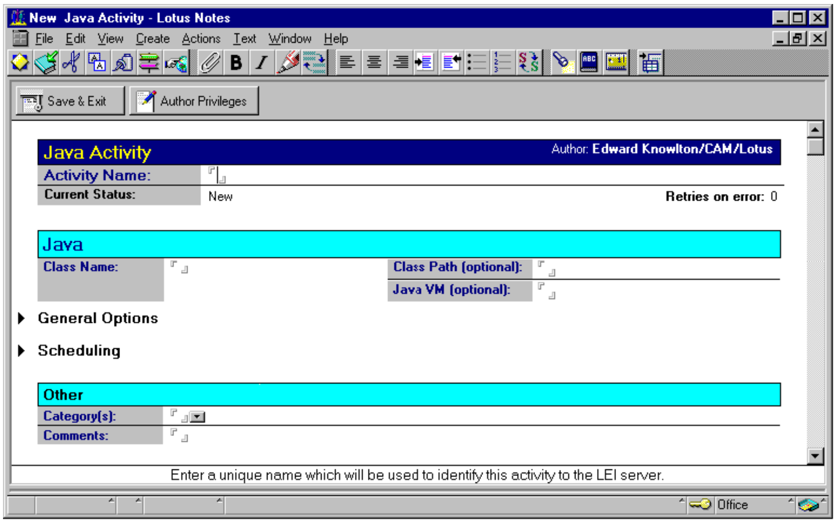
## Java Activity Document

The Java Activity document specifies the information required to run an LEI Java application.

See Chapter 17, "Introduction to LEI Activities," for descriptions of the following sections, which are common to all documents:

- Activity Name and Status
- Activity Options
- Scheduling

Choose Create – Activity – Java or click Create Activity in the Navigator and select Java to access the Java Activity document shown below.


New Java Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Office

Java Activity Author: Edward Knowlton/CAM/Lotus

Activity Name:

Current Status: New Retries on error: 0

Java

Class Name:

Class Path (optional):

Java VM (optional):

General Options

Scheduling

Other

Category(s):

Comments:

Enter a unique name which will be used to identify this activity to the LEI server.

## Java

Use this section of the Java Activity document to identify the main Java Class and optionally the CLASSPATH and Virtual Machine that the Activity will use.

| *Field* | *Required* | *Description* |
|---|---|---|
| Class Name: | Yes | Name of the Java Class to execute, with any command line parameters. |
| Class Path: | Optional | This is optional. Any entry here will prepend the existing CLASSPATH environment variable set on the Server that will execute the Java application. |
| Java VM: | Optional | Optional Java Virtual Machine override. If left blank, java.exe will execute.<br>You may enter the path to an alternate JVM to execute the Java application. Doing so may require a change in the CLASSPATH. |

# Chapter 28
# Scripted Activity

This chapter provides information about the LEI Scripted Activity. Included is information about what this Activity is, when and why to use it, and how to create the Activity.

## Introduction to the Scripted Activity

A Scripted Activity is one that executes LotusScript commands. LEI extends the set of Notes product classes available to LotusScript. The Domino Connector LSX, which comes with Notes/Domino 4.63 and later, is also included with LEI. LEI extends these classes with additional methods to support the LEI Administrator and log databases. Using LEI LotusScript classes allows you extend the functionality available from the other LEI Administrator Activity operations (Direct Transfer, Polling, Replication, etc.). If you are upgrading from NotesPump, please see, “If You are Migrating from NotesPump” in this chapter.

### When to Use the Scripted Activity

Use the LotusScript LEI classes to construct customized routines in situations where you need more complete control over source and destination data transfers, or wish to instruct the LEI engine to perform data massaging or evaluation during transfer.

## How to Create a Scripted Activity

Follow these steps to create a LEI LotusScript Activity.

1. Install the Development Client to a Notes R4.6 Client or Domino R5.0x Designer machine to access the LEI LotusScript classes.
2. From the Lotus Notes R4.6x or Domino R5.0x Designer database Design Agents editor window (also known as the Integrated Development Environment or IDE) type the following in the options section of the script window.:

   ```
   Uselsx "*lsxlei"
   ```

3. Within the IDE, create the LotusScript routine using Domino Connector classes.

   **Note** Once you save the script and run it once, the system loads the LEI LSX and the descriptions. The Lotus Connector classes will then appear automatically in the Notes Classes section of the script browser drop down list.

4. Within the LEI Administrator – Scripted Activity, identify the Lotus Notes database and server where the LotusScript routine has been stored. Proceed to schedule and execute the LEI Scripted Activity.

**Note** Scripted agents written with the LEI LSX differ from those written with the LC LSX. Scripts written for LEI read their Connector configuration information from the LEI Administrator and log their results to the LEI Log. Also, scripts written for the LEI LSX must begin with the following statement:

```
Uselsx "*lsxlei"
```

See the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for more information.

Script development requires that either an LEI Development Client and/or the LEI Server be installed. With LEI installed, scripts must read their Connection information from the LEI Administrator.

Using the correct lei.ini file ensures that the Domino Connector classes are initialized correctly. The lei.ini file is documented in the “Introduction to Installing and Configuring Lotus Enterprise Integrator” chapter of the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide*.

The use of the LEI Administrator and logging is established when the LCSession class object is created. LEI Scripted Activities require that the first Domino Connector Class object created be an LCSession object. The LCSession object must be created with a name.

**Note** With scripted Activities, the Activity name is used in the log. The session name is only used in the log when the Activity is run manually (not as part of a scripted Activity).

The use of the LEI Administrator also changes the behavior of the LCConnection class. LEI requires that the name given when creating each LCConnection object be the name of a Connection document from the LEI Administrator. The Connection parameters are loaded from the corresponding Connection document. By contrast, LC LSX requires that the name given when creating each LCConnection object be the Connector's library name and all parameters must be assigned explicitly.

Using the LEI LSX, the session/Activity is created with a name and called with a valid text string. Connections are then created using a Connection name from the LEI Administrator.

When a scripted agent is called, it is calling a defined Activity. This Activity must have named Connection documents associated with it. As a result, when the LCConnection object is instantiated with the following call, the <libraryname> takes on a new meaning within this context. It becomes the name of a Connection to be used by the script.

```
Dim <connectionname> as New LCConnection ( <libraryname> )
```

See Appendix E of the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for details on LEI-specific behaviors, including description of the LCSession and LCConnection classes. Also see the section entitled "How LEI Supports LotusScript" later in this chapter.

## How LEI Supports LotusScript

The LEI LotusScript is first created as an agent using the Notes Integrated Development Environment (IDE), and later LEI executes the agent. The Lotus Domino Connector LSX extends the LotusScript language to support the complete LEI programming interface. The environment for developing an LEI scripted agent requires that both Notes and LEI be installed on the development machine. When editing a LotusScript agent, the Notes IDE loads the Domino Connector LSX. This permits the LotusScript syntax to be verified. When the agent is executed or debugged, the LSX establishes a connection with the local LEI Server or Client to process database Connections and perform logging.

To use the LEI extensions to LotusScript, include the Domino Connector LSX within a scripted agent. This is accomplished with the UseLSX statement. The LEI installation registers its LSX so that the following platform independent syntax may be used within a scripted agent's 'Options' section:

```
Uselsx "*lsxlei"
```

Additionally, both Notes and LEI need their program directories on the system path. Notes requires the LEI program directory to be on the path so the IDE may load the LSX for authoring and testing of the agent; LEI requires the Notes program directory to be on the system path so that the agent may be executed as part of the scripted Activity.

See Appendix E of the *Lotus Enterprise Integrator Domino Connector LotusScript Extensions Guide* for details on LEI-specific behaviors, including description of the LCSession and LCConnection classes. Also see the previous section entitled "How to Create a Scripted Activity".

## The Scripted Activity Document

The Scripted document includes the name of the Notes agent that contains the LotusScript to be executed, as well as the standard scheduling options common to all Activities.

To see the agent, select the Notes database which contains the agent (the default is the LEI Script Vault). Next, choose View – Design from the Notes Menu and then select Agents in the Navigator.

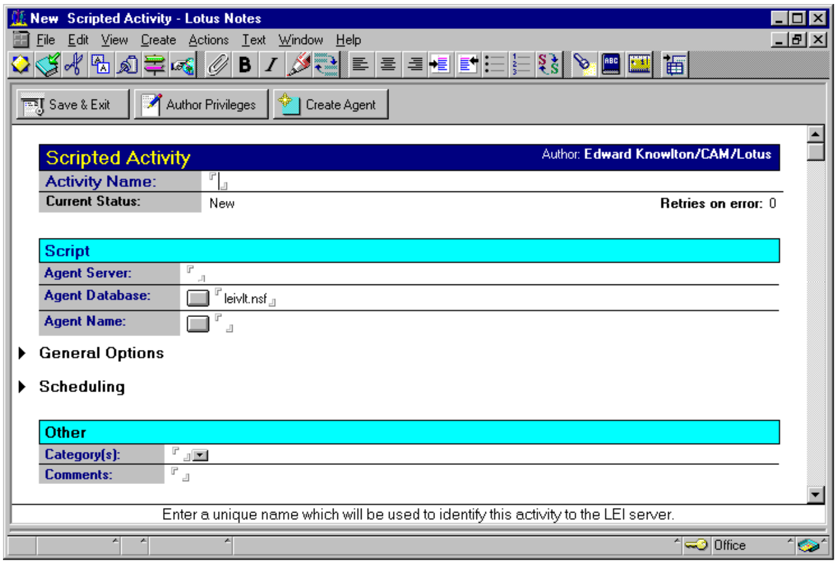
See Chapter 17, "Introduction to LEI Activities", for descriptions of the following sections of the Scripted Activity document, which are common to all Activity documents:

- Activity Name and Status
- General Options
- Scheduling

The fields unique to the Scripted Activity are described in the following sections.

**Note** There is a complete example of building a Scripted Activity in Appendix E, Tutorials.

The Scripted Activity document, shown below, appears when you click Create Activity in the navigator and select Scripted, or choose Create – Activity – Scripted in the LEI Administrator. Use this document to create a Scripted Activity.


New Scripted Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Create Agent

Scripted Activity — Author: Edward Knowlton/CAM/Lotus

Activity Name:

Current Status: New — Retries on error: 0

Script

Agent Server:

Agent Database: leivlt.nsf

Agent Name:

General Options

Scheduling

Other

Category(s):

Comments:

Enter a unique name which will be used to identify this activity to the LEI server.

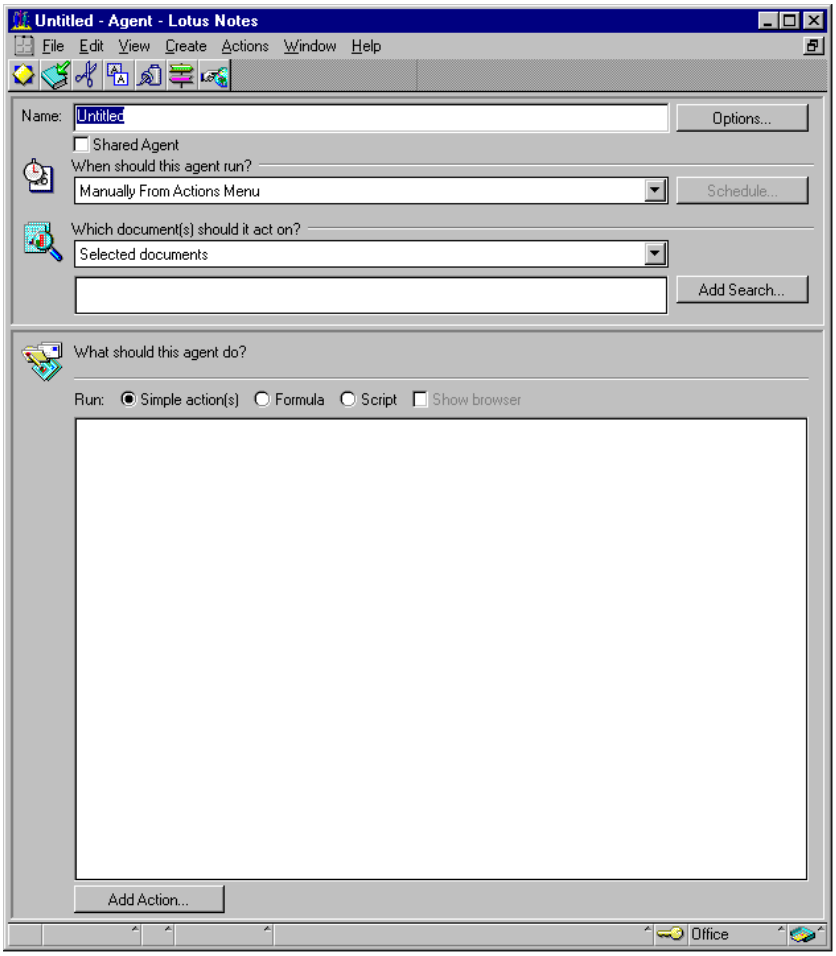
## Script

This section identifies the LotusScript that the Activity will use.

| *Field* | *Description* |
|---|---|
| Agent Server | Server where the agent containing the script is located. |
| Agent Database | Notes database file path and name where the agent containing the script is found. Note: The Agent Database browse button located next to the field displays the list of databases available on the server chosen above. A choice can be made from this list and it will be inserted into the field. |
| Agent Name | Name of the agent containing the script to execute. **Note** The Agent browser button located next to the field displays the list of agents available on the database selected above. Choosing from the list will insert the selection into the field. |

## Agent Development Document

Shown below is the Notes Agent Development document. Click the Create Agent action button in the Scripted Activity document to access this document. Use this document to create your LotusScript agent.


Untitled - Agent - Lotus Notes

Name: Untitled

Shared Agent

When should this agent run? Manually From Actions Menu

Which document(s) should it act on? Selected documents

What should this agent do?

Run: Simple action(s) Formula Script Show browser

Notes online documentation describes the procedure for creating a Notes agent. Please see the Notes help system to learn how to perform that task.

To check to see if an agent is running, type the following Show Task command line entry in the Domino server console:

```
sh ta
```

One of the processes listed will be the Agent Manager and it will display as either idle or the name of the task that it is running. To check the log after the script has run, go to the database that it is stored in, select the agent and from the menu at the top, choose Agent – Log. The log is overwritten each time the agent runs. For each run it will state the number of documents selected and the time it started and stopped.

## Notes about Creating Scripted Activities

Here are a few suggestions on writing Notes Agents to be used in LEI Scripted Activities.

- You will need to have both the directories for LEI and Notes on your path to allow LotusScript to execute correctly.
- For all agents you create, use the following settings:
  - Make the Agent shared
  - Select "Manually from the Agent List"
  - Select "Run Once"
- It is a good idea to provide support for both LEI errors and LotusScript errors.
- To use the LEI extensions to LotusScript, you will need to have the following statement in your agent's (Options) section:

  ```
  Uselsx "*lsxlei"
  ```

- You may test Agents through the Notes Integrated Development Environment (IDE). Select "Debug LotusScript" from the File · Actions menu. Then Run the agent from the menu.

  **Note** You will need either a LEI Server or Client installation on the local machine.

  In addition, Notes password entry is not permitted. Either a Notes ID without a password must be used or the option "Share password with Notes add-ins." Select the Notes menu option File - Tools - User ID.

- You may choose to organize your agents within the LEI Script Vault database, the Notes databases where related data resides, or collected in some other database.

  **Caution** Since these agents are "shared", they will be deleted if the database design is replaced or refreshed. Lotus recommends that you disable the design inheritance of the Notes database where you choose to store the agents.

## Migrating from Notes Pump

The Domino Connector Classes LotusScript Extension library (LSX) that ships with Lotus Enterprise Integrator (LEI) has changed from the LSX that shipped with NotesPump. This change will affect any script that was written with the NotesPump LSX (*nplsx.dll) and is being run as a scripted Activity by NotesPump. Migrating your Administrator database using LEI's setup will not upgrade your scripts nor are scripts written with the NP LSX able to run with the LSX that ships with LEI. This also means that scripts written using the LEI LSX will not run under NotesPump.

The LSX that ships with LEI (*lsxlei.dll) is the Domino Connector Classes as shipped with Domino but with added advanced capabilities. This means that if you install or upgrade to Domino 4.63x or Domino 5.0 after you install LEI, you must reinstall LEI to ensure that you have the Domino Connector Classes with these advanced features.

Please review your Domino Connector Classes Guide to familiarize yourself with what has changed. To enable your scripts to run using the LSX provided with LEI, follow the steps outlined below.

1. Replace "*lnplsx" with "*lsxlei" in Options.
2. Look at declarations and replace any references to "NPActivity" with "LCSession". Only one parameter is needed for LCSession and that is a name which is of type string.
3. Search and replace all instances of "NP" with "LC" (you may want to do it instance by instance if any variables are prefixed with NP)
4. Search and replace all instances of "LCConnect" (gotten as a result of doing the step above so it was formerly NPConnect) with "LCConnection", remove first parameter (called a LinkToken in the NP LSX) since it is no longer needed.
5. Replace all NPactivity objects with LCSession objects – however you will need to look at the new methods under LCSession to find which method is the one you want to use since many names and parameters have changed (for example NPactivity.LogStream might be replaced with LCSession.Logtext)
6. Replace all .asLSString references with .text.Replace all .asLSXXXX where XXXX is LSLong, LSDouble etc. with .value.
7. In many cases in the NP LSX, the last parameter was count (number of records affected by the operation) – this is now the return value of many functions such as LCConnection.Fetch, LCConnection.Execute, LCConnection.Insert, LCConnection.Update etc.

The instructions below can be done in any order.

Go through each function call, making sure that parameters match the new specifications. If NPFieldList.Merge was used, it will convert (to LCFieldList.Merge)and work fine – however there are two new methods (LCFieldList.Map, LCFieldList.MapName) that make accomplishing field mapping much easier.

**Note** Activity.RaiseExceptions no longer exists as a property (there is no LCSession.RaiseExceptions). Formerly this property could be toggled to raise exceptions to LotusScript or not. Now all error handling is done the way LotusScript itself works – if there is an error, it will be raised. You are given the choice of how to handle it using conventional LotusScript types of error handling. Please see the *LotusScript Language Reference* for more details on LotusScript error handling.

## Correspondence between NPLSX and Lotus Connector LSX

In the following chart, NotesPump (NP) functions that are in bold type indicate functions that have no counterpart in the LC LSX (LEI LSX).

The following conversion chart shows the correspondence between the NPLSX and the LSX.

### NPActivity Class and LCSession Class

| *NPActivity Class NP Method* | *NP Syntax* | *LCSession Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NPActivity constructor | | New | Dim variableName As New LCSession (“SessionName”) |
| AddProperty | AddProperty (npPropertyName As Variant, ByVal npPropertyStorage As NPINT, npPropertyToken As NPINT) As NPSTATUS | AddProperty | tokenID = lcSession.AddProperty (propertyName, storageType) |
| ClearStatus | ClearStatus () | ClearStatus | thisSession. ClearStatus () |
| GetCommand | GetCommand () As NPSTATUS | GetCommand | GetCommand () As NPSTATUS |

*continued*

| *NPActivity Class NP Method* | *NP Syntax* | *LCSession Class LC Method* | *LC Syntax* |
|---|---|---|---|
| GetLogLast | GetLogLast (npStream As NPSTREAM, npExternalCode As NPINT, npExternalStream As NPSTREAM) As NPSTATUS | | |
| GetProperty | GetProperty (ByVal npPropertyToken As NPINT, npDestField As NPFIELD) As NPSTATUS | GetProperty | destField = lcSession.GetProperty (propertyToken) |
| GetProperty <Type> | GetPropertyBoolean (ByVal npPropertyToken As NPINT, ByVal npDefault As NPBOOL) As NPBOOL GetProperty Currency (ByVal npPropertyToken As NPINT, npDestCurrency As NPCURRENCY) As NPSTATUS GetPropertyDatet | GetProperty <Type> | flag = lcSession.GetPropertyBoolean(propertyToken, default) Set destCurrency = lcSession.GetProperty Currency(propertyToken) Set destDatetime = lcSession.GetProperty Datetime(propertyToken) destFloat = lcSession.GetProperty Float(propertyToken) destInt = lcSession.GetProperty Int(propertyToken) Set destNumeric = lcSession.GetProperty Numeric(propertyToken) Set destStream = lcSession.GetProperty Stream(propertyToken, streamFormat) |
| GetStatus | GetStatus () As NPSTATUS | GetStatus | Call thisSession. GetStatus (statusText, externalCode, externalText) |

*continued*

| *NPActivity Class NP Method* | *NP Syntax* | *LCSession Class LC Method* | *LC Syntax* |
|---|---|---|---|
| GetStatus AsLSString | GetStatusAsLSString (ByVal npStatus As NPSTATUS) As String | GetStatusText | message = thisSession. GetStatusText (errorCode) |
| GetStatus Stream | GetStatusStream (ByVal npStatus As NPSTATUS, npStream As NPSTREAM) As NPSTATUS | | |
| ListLink | ListLink (ByVal npFirst As NPINT, npLinkCode As NPINT, npLinkLibrary As NPSTREAM) As NPSTATUS | ListConnector | Call thisSession. ListConnector (List, ConnectorName, ConnectorCode, identifyFlagList, identifyNameList) |
| | | ListMeta Connector | Call thisSession. ListMetaConnector (list, metaconnectorName, ConnectorCode, identifyFlagList, identifyNameList) |
| ListProperty | ListProperty (ByVal npFirst As NPBOOL, npPropertyToken As NPINT, npDataType As NPTYPE, npPropFlags As NPFLAGS, npPropertyName As NPSTREAM, npReserved1 As NPINT) As NPSTATUSA | ListProperty | Call lcSession.ListProperty (list, propertyToken, dataType, property Flags, propertyName) |
| Log | Log (ByVal npStatus As NPINT) As NPSTATUS | Log | lcSession.Log (status) |

*continued*

| *NPActivity Class NP Method* | *NP Syntax* | *LCSession Class LC Method* | *LC Syntax* |
|---|---|---|---|
| LogEx | LogEx (ByVal npStatus As NPINT, ByVal npInt1 As NPINT, ByVal npInt2 As NPINT, ByVal npInt3 As NPINT, npStream1 As NPSTREAM, npStream2 As NPSTREAM, npStream3 As NPSTREAM) As NPSTATUS | LogEx | lcSession.LogEx (status, int1,int2, int3, text1, text2, text3) |
| LogStream | LogStream (npLogStreamFlags As NPFLAGS, npExternalStream As NPSTREAM, ByVal npExternalCode As NPINT) As NPSTATUS | LogText | lcSession.LogText(logFlags, text, externalCode) |
| LogStreamEx | LogStreamEx (npLogStreamFlags As NPFLAGS, npExternalStream As NPSTREAM, ByVal npExternalCode As NPINT, ByVal npInt1 As NPINT, ByVal npInt2 As NPINT, ByVal npInt3 As NPINT, S | LogTextEx | lcSession.LogTextEx(logFlags, text, externalCode, int1, int2, int3, text1, text2, text3) |
| LookupLink | LookupLink (npLinkLibrary As Variant, npLinkCode As NPINT) As NPSTATUS | Lookup Connector | Call thisSession. LookupConnector (ConnectorName, ConnectorCode, identifyFlagList, identifyNameList) |

*continued*

| *NPActivity Class NP Method* | *NP Syntax* | *LCSession Class LC Method* | *LC Syntax* |
|---|---|---|---|
| | | Lookup MetaConnector | Call thisSession. LookupMetaConnector (metaconnectorName, ConnectorCode, tokenBase, identifyFlagList, identifyNameList) |
| | | Lookup Property | flag = lcSession.Lookup Property (propertyToken, dataType, propertyFlags, propertyName) |
| Print | Print (npMessage As Variant) As NPSTATUS | | |
| RunActivity | RunActivity (npActivityList As Variant, ByVal npActRunFlags As NPFLAGS, npStopDatetime As NPDATETIME) As NPSTATUS | RunActivity | Call thisSession.Run Activity (name, runFlags, stopDatetime) |
| SetProperty | SetProperty (ByVal npPropertyToken As NPINT, npSrcField As NPFIELD) As NPSTATUS | SetProperty | Call thisSession.SetProperty (propertyToken, srcField) |

*continued*

| *NPActivity Class NP Method* | *NP Syntax* | *LCSession Class LC Method* | *LC Syntax* |
|---|---|---|---|
| SetProperty <Type> | SetProperty Currency (ByVal npPropertyToken As NPINT, npSrcCurrency As NPCURRENCY) As NPSTATUS SetProperty Datetime (ByVal npPropertyToken As NPINT, npSrcDatetime As NPDATETIME) As NPSTATUS SetPropertyFt | SetProperty <Type> | Call lcSession.SetProperty Boolean(propertyToken, srcBoolean) Call lcSession.SetProperty Currency(propertyToken, srcCurrency) Call lcSession.SetProperty Datetime(propertyToken, srcDatetime) Call lcSession.SetProperty Float(propertyToken, srcFloat) Call lcSession.SetPropertyInt (propertyToken, srcInt) Call lcSession.SetProperty Numeric(propertyToken, srcNumeric) Call lcSession.SetProperty Stream(propertyToken, srcStream) |
| | | Sleep | Call thisSession. Sleep (milliSeconds) |

## NPConnect Class and LCConnection Class

| *NPConnect Class NP Method* | *NP Syntax* | *LCConnection Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NPConnect constructor | New (ByVal npLinkToken As NPINT, npLinkName As Variant) As NPConnect | New | Dim connectionName as New LC Connection (libraryName) |
| NPConnect destructor | | | |
| Action | Action (ByVal npActionType As NPINT) As NPSTATUS | Action | Call lcConnection. Action (actionType) |
| | | Call | count = lcConnection. Call (parmFieldList, recordIndex, destFieldList) |

*continued*

| *NPConnect Class NP Method* | *NP Syntax* | *LCConnection Class LC Method* | *LC Syntax* |
|---|---|---|---|
| Catalog | Catalog (ByVal npObjectType As NPINT, npDestFieldlist As NPFIELDLIST, npAffectedCount As NPINT) As NPSTATUS | Catalog | |
| Connect | Connect () As NPSTATUS | Connect | lcConnection. Connect |
| | | Copy | Set newConnection = lcConnection. Copy() |
| Create | Create (ByVal npObjectType As NPINT, npSrcFieldlist As NPFIELDLIST) As NPSTATUS | Create | Call lcConnection. Create (objectType, srcFieldlist ) |
| Disconnect | Disconnect () As NPSTATUS | Disconnect | lcConnection. Disconnect |
| Drop | Drop (npObjectType As NPINT) As NPSTATUS | Drop | Call lcConnection. Drop (objectType) |
| Execute | Execute (npStatement As Variant, npDestFieldlist As NPFIELDLIST, npAffectedCount As NPINT) As NPSTATUS | Execute | count = lcConnection. Execute (statement, destFieldlist) |
| Fetch | Fetch (npDestFieldlist As NPFIELDLIST, ByVal npRecordIndex As NPINT, ByVal npRecordCount As NPINT, npAffectedCount As NPINT) As NPSTATUS | Fetch | count = lcConnection. Fetch (destFieldlist, recordIndex, recordCount) |
| GetProperty | GetProperty (ByVal npPropertyToken As NPINT, npDestField As NPFIELD) As NPSTATUS | GetProperty | Call thisConnection. GetProperty (propertyToken, destField) |

*continued*

| *NPConnect Class NP Method* | *NP Syntax* | *LCConnection Class LC Method* | *LC Syntax* |
|---|---|---|---|
| GetProperty <Type> | GetPropertyBoolean (ByVal npProperty Token As NPINT, ByVal npDefault As NPBOOL) As NPSTATUS GetProperty Currency (ByVal npPropertyToken As NPINT, npDestCurrency As NPCURRENCY) As NPSTATUS GetPropertyDay | GetProperty <Type> | flag = lcConnection. GetProperty Boolean (propertyToken, default) Set destCurrency = lcConnection. GetProperty Currency (propertyToken) Set destDatetime = lcConnection. GetProperty Datetime (propertyToken) destFloat = lcConnection. GetPrope |
| Insert | Insert (npSrcFieldlist As NPFIELDLIST, ByVal npRecordIndex As NPINT, ByVal npRecordCount As NPINT, npAffectedCount As NPINT) As NPSTATUS | Insert | count = lcConnection. Insert (srcFieldlist, recordIndex, recordCount) |
| ListProperty | ListProperty (ByVal npFirst As NPBOOL, npPropertyToken As NPINT, npDataType As NPTYPE, npPropFlags As NPFLAGS, npPropertyName As NPSTREAM) As NPSTATUS | ListProperty | Call lcConnection. listProperty (list, propertyToken, dataType, propertyFlags, propertyName) |
| | | Lookup Property | flag = lcConnection. Lookup Property(propertyT oken, dataType, propertyFlags, propertyName) |

*continued*

| *NPConnect Class NP Method* | *NP Syntax* | *LCConnection Class LC Method* | *LC Syntax* |
|---|---|---|---|
| Remove | Remove (npKeyFieldlist As NPFIELDLIST, ByVal npRecordIndex As NPINT, ByVal npRecordCount As NPINT npAffectedCount As NPINT) As NPSTATUS | Remove | count = lcConnection. Remove (keyFieldlist, recordIndex, recordCount) |
| Select | Select (npKeyFieldlist As NPFIELDLIST, ByVal npRecordIndex As NPINT, npDestFieldlist As NPFIELDLIST, npAffectedCount As NPINT) As NPSTATUS | Select | count = lcConnection. Select (keyFieldlist, recordIndex, destFieldlist) |
| SetProperty | SetProperty (ByVal npPropertyToken As NPINT, npSrcField As NPFIELD) As NPSTATUS | SetProperty | Call thisConnection. SetProperty (propertyToken, srcField) |
| SetProperty <Type> | SetPropertyCurrency (ByVal npPropertyToken As NPINT, npSrcCurrency As NPCURRENCY) As NPSTATUS SetPropertyDatetime (ByVal npPropertyToken As NPINT, npSrcDatetime As NPDATETIME) As NPSTATUS SetPropertyFy | SetProperty <Type> | Call lcConnection. SetPropertyBoolean (propertyToken, srcBoolean) Call lcConnection. SetProperty Currency (propertyToken, srcCurrency) Call lcConnection. SetProperty Datetime (propertyToken, srcDatetime) Call lcConnection. SetPropertyFlo |

*continued*

| *NPConnect Class NP Method* | *NP Syntax* | *LCConnection Class LC Method* | *LC Syntax* |
|---|---|---|---|
| Update | Update (npSrcFieldlist As NPFIELDLIST, ByVal npRecordIndex As NPINT, ByVal npRecordCount As NPINT, npAffectedCount As NPINT) As NPSTATUS | Update | count = lcConnection. Update (srcFieldlist, recordIndex, recordCount) |

## NPCurrency Class and LCCurrency Class

| *NPCurrency Class NP Method* | *NP Syntax* | *LCCurrency Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NPCurrency constructor | New () As NPCURRENCY | New | Dim variableName as New LCCurrency |
| Add | Add (npCurrency1 As NPCURRENCY, npCurrency2 As NPCURRENCY) As NPSTATUS | Add | Call currencyTotal. Add (currency1, currency2) |
| Clear | Clear () | | |
| Compare | Compare (npBaseCurrency As NPCURRENCY, npResult As NPINT) As NPSTATUS | Compare | Result = thisCurrency. Compare (baseCurrency) |
| Copy | Copy (npSrcCurrency As NPCURRENCY) | Copy | Set newCurrency = srcCurrency. Copy |
| FromFloat | FromFloat (ByVal npSrcFloat As NPFLOAT) As NPSTATUS | | |
| FromStream | FromStream (npSrcStream As NPSTREAM) As NPSTATUS | | |
| Subtract | Subtract (npCurrency1 As NPCURRENCY, npCurrency2 As NPCURRENCY) As NPSTATUS | Subtract | Call currencyResult. Subtract (currency1, currency2) |

## NPDatetime Class and LCDatetime Class

| *NPDatetime Class NP Method* | *NP Syntax* | *LCDatetime Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NPDatetime constructor | New () As NPDATETIME | New | Dim variableName as New LCDatetime (Year, Month, Day, Hour, Minute, Second, Hundredth, Zone, DST) |
| Adjust | Adjust (npSrcDatetimeParts As NPDATETIMEPARTS) As NPSTATUS | Adjust | Call changeDatetime. Adjust (Units, Amount) |
| Clear | Clear As NPSTATUS | Clear | Call dateTimeObject. Clear |
| Compare | Compare (npBaseDatetime As NPDATETIME, npResult As NPINT) As NPSTATUS | Compare | Result = thisDatetime. Compare (baseDatetime) |
| Copy | Copy (npSrcDatetime As NPDATETIME) | Copy | Set newDatetime = srcDatetime. Copy |
| FromStream | FromStream (npSrcStream As NPSTREAM) As NPSTATUS | | |
| GetDiff | GetDiff (npBaseDatetime As NPDATETIME, ByVal npDatetimeUnit As NPINT, npDifference As NPINT) As NPSTATUS | GetDiff | difference = lcDatetime. GetDiff (baseDatetime, Units) |

## NPDateTimeParts Class

| *NPDateTmeParts Class NP Method* | *NP Syntax* | *LC Method* | *LC Syntax* |
|---|---|---|---|
| NPDatetimeParts constructor | New () As NPDatetimeParts | | |
| GetJulian | GetJulian () As NPINT | | |
| GetParts | Get (ByVal npNullAsZero As NPINT, npDestDatetimeParts As NPDATETIMEPARTS) As NPSTATUS | | |
| GetTicks | GetTicks () As NPINT | | |
| SetConstant | SetConstant (ByVal npDatetimeConstant As NPINT) As NPSTATUS | SetConstant | Call lcDatetime. SetConstant (datetimeConstant) |
| SetCurrent | SetCurrent () As NPSTATUS | SetCurrent | Call lcDatetime Object. SetCurrent |
| SetJulian | SetJulian (ByVal Julian As NPINT) As NPSTATUS | | |
| SetParts | SetParts (npSrcDatetimeParts As NPDATETIMEPARTS, ByVal npUseLocalZone As NPBOOL) As NPSTATUS | | |
| SetTicks | SetTicks (ByVal npTicks As NPINT) As NPSTATUS | | |

## NPFIELD Class and LCFIELD Class

| *NPFIELD Class NP Method* | *NP Syntax* | *LCFIELD Class LC Method* | *LC Syntax* |
|---|---|---|---|
| | | new | Dim variablename as New LCField( type, count ) |
| NPField constructor | New (ByVal npDataType As NPTYPE, ByVal npValueCount As NPINT) As NPFIELD | ClearVirtual Code | Call field. ClearVirtualCode (code) |
| Compare | Compare (ByVal npThisIndex As NPINT, npBaseField As NPFIELD, ByVal npBaseIndex As NPINT, ByVal npValueCount As NPINT, ByVal npLowPrecision As NPBOOL, npResult As NPINT) As NPSTATUS | Compare | Result = thisField. Compare (thisIndex, baseField, baseIndex, valueCount, compareFlags) |
| Convert | Convert (ByVal npThisIndex As NPINT, npSrcField As NPFIELD, ByVal npSrcIndex As NPINT, ByVal npValueCount As NPINT) As NPSTATUS | Convert | Call thisField. Convert (thisIndex, srcField, srcIndex, valueCount) |
| Copy | Copy (npSrcField As NPFIELD) As NPSTATUS | Copy | Set newField = origField. Copy |
| GetBuffer | GetBuffer (ByVal npIndex As NPINT, npReserved1 As Variant, npReserved2 As Variant) As NPSTATUS | | |

*continued*

| *NPFIELD Class NP Method* | *NP Syntax* | *LCFIELD Class LC Method* | *LC Syntax* |
|---|---|---|---|
| GetFlags | GetFlags () As NPFLAGS | | |
| GetFormat Datetime | GetFormatDatetime (npDatetimeFlags As NPFLAGS, npSize As NPINT) As NPSTATUS | GetFormat Datetime | Call thisField. GetFormatDatetime (dateTimeFlags, size) |
| GetFormat Number | GetFormatNumber (npNumberFlags As NPFLAGS, npSize As NPINT, npPrecision As NPINT, npScale As NPINT) As NPSTATUS | GetFormat Number | Call thisField. GetFormatNumber (numberFlags, size, precision, scale) |
| GetFormat Stream | GetFormatStream (npStreamFlags As NPFLAGS, npMaxLength As NPINT, npStreamFormat As NPINT) As NPSTATUS | GetFormat Stream | Call thisField. GetFormatStream (streamFlags, maxLength, streamFormat) |
| GetType | GetType () As NPTYPE | | |
| GetTypeSize | GetTypeSize () As NPINT | | |
| GetValueCount | GetValueCount () As NPINT | | |
| GetVirtualCode | GetVirtualCode () As NPINT | | |

*continued*

| *NPFIELD Class NP Method* | *NP Syntax* | *LCFIELD Class LC Method* | *LC Syntax* |
|---|---|---|---|
| Get<Type> | GetCurrency (ByVal npIndex As NPINT, npDestCurrency As NPCURRENCY, npIsNull As NPBOOL) As NPSTATUS GetDatetime (ByVal npIndex As NPINT, npDestDatetime As NPDATETIME, npIsNull As NPBOOL) As N | | |
| | | GetCurrency | Set newCurrency = lcField. GetCurrency (index) |
| | | GetDatetime | Set newDatetime = lcField. GetDatetime (index) |
| | | GetFieldlist | Set newFieldList = lcField. GetFieldList (index) |
| | | GetFloat | Set newFloat = lcField. GetFloat (index) |
| | | GetInt | Set newInt = thisField. GetInt (index) |
| | | GetNumeric | Set newNumeric = thisField. GetNumeric (index) |
| | | GetStream | Set newStream = thisField. GetStream (index, streamFormat) |
| IsNull | IsNull (ByVal npIndex As NPINT) As NPBOOL | | |
| | | LookupVirtual Code | Flag = field. LookupVirtualCode (virtualCode) |

*continued*

| *NPFIELD Class NP Method* | *NP Syntax* | *LCFIELD Class LC Method* | *LC Syntax* |
|---|---|---|---|
| SetFlags | SetFlags (ByVal FieldFlags As NPFLAGS) As NPSTATUS | | |
| SetFormat Datetime | SetFormatDatetime (ByVal DatetimeFlags As NPFLAGS, ByVal npSize As NPINT) As NPSTATUS | SetFormat Datetime | Call thisField. SetFormatDatetime (datetimeFlags, size) |
| SetFormat Stream | SetFormatStream (ByVal npStreamFlags As NPFLAGS, ByVal npMaxLength As NPINT, ByVal npStreamFormat As NPINT) As NPSTATUS | SetFormat Stream | Call thisField. SetFormatStream (streamFlags, maxLength, streamFormat) |
| SetFormat Number | SetFormatNumber (ByVal npNumberFlags As NPFLAGS, ByVal npSize As NPINT, ByVal Precision As NPINT, ByVal npScale As NPINT) As NPSTATUS | SetFormat Number | Call thisField. SetFormatNumber (numberFlags, size, precision, scale) |
| SetNull | SetNull (ByVal npIndex As NPINT, ByVal npIsNull As NPBOOL) As NPSTATUS | | |
| SetVirtualCode | SetVirtualCode (ByVal npVirtualCode As NPINT) As NPSTATUS | SetVirtual Code | Call thisField. SetVirtualCode (virtualCode) |

*continued*

| *NPFIELD Class NP Method* | *NP Syntax* | *LCFIELD Class LC Method* | *LC Syntax* |
|---|---|---|---|
| Set<Type> | SetCurrency (ByVal npIndex As NPINT, npSrcCurrency As NPCURRENCY) As NPSTATUS SetDatetime (ByVal npIndex As NPINT, npSrcDatetime As NPDATETIME) As NPSTATUS SetFloat (ByVal npIndex As NPINT, ByVal npSrcFloatl | | |
| | | SetCurrency | Call thisField.SetCurrency (index, srcCurrency) |
| | | SetDateTime | Call thisField.SetDatetime (index, srcDatetime) |
| | | SetFieldList | Call thisField.SetFieldlist (index, srcFieldlist) |
| | | SetFloat | Call thisField.SetFloat (index, srcFloat) |
| | | SetInt | Call thisField. SetInt (index, srcInt) |
| | | SetNumeric | Call thisField. SetNumeric (index, srcNumeric) |
| | | SetStream | Call thisfield. SetStream (index, srcStream) |

## NPFieldlist Class and LCFieldlist Class

| *NPFieldlist Class NP Method* | *NP Syntax* | *LCFieldlist Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NPFieldlist constructor | New (ByVal npRecordCount As NPINT, ByVal Flags As NPFLAGS) As NPFIELDLIST | | Dim variableName as New LCFieldlist (recordCount, fieldFlags) |
| Append | Append (npFieldName As Variant, ByVal npDataType As NPTYPE, npNewField As Variant) As NPSTATUS | Append | Set field = fieldlist. Append (fieldName, dataType) |
| Copy | Copy (npSrcFieldlist As NPFIELDLIST) As NPSTATUS | Copy | Set fldListRecord New = fldList Record. Copy |
| | | CopyField | Set newField = fieldList. CopyField (index, srcField, name) |
| CopyRef | CopyRef (npSrcFieldlist As NPFIELDLIST) As NPSTATUS | CopyRef | Set fldListRecord New = fldList Record. CopyRef |
| GetCount | GetCount () As NPINT | | |
| GetField | GetField (ByVal npIndex As NPINT, ByVal npStreamFormat As NPINT, npDestField As Variant, npReserved1 As Variant) As NPSTATUS | GetField | Set field = fldLstRecord. GetField ( index ) |
| GetName | GetName (ByVal npIndex As NPINT, ByVal npStreamFormat As NPINT, npFieldName As Variant) As NPSTATUS | GetName | fieldName = fieldListRecord. GetName (index) |
| GetNameLength | GetNameLength (ByVal npStreamFormat As NPINT) As NPINT | | |
| GetRecordCount | GetRecordCount () As NPINT | | |
| GetSequence | GetSequence () As NPINT | | |

*continued*

| *NPFieldlist Class NP Method* | *NP Syntax* | *LCFieldlist Class LC Method* | *LC Syntax* |
|---|---|---|---|
| | | IncludeField | Call fldLstRecord. IncludeField (index, field, fieldName) |
| Insert | Insert (ByVal npIndex As NPINT, npFieldName As Variant, ByVal npDataType As NPTYPE, npNewField As Variant) As NPSTATUS | Insert | Set fieldNew = fldLstRecord. Insert (index, fieldName, dataType) |
| List | List (npDestField As Variant, npIndex As NPINT, npDataType As NPTYPE, FieldFlags As NPFLAGS npReserved1 As Variant) As NPSTATUS | List | Call fldLstRecod. List (position, destField, index, dataType, flags, fieldName) |
| ListSetup | ListSetup (ByVal npStreamFormat As NPINT) As NPSTATUS | | |
| Lookup | Lookup (npFieldName As Variant, npDestField As Variant, npIndex As NPINT) As NPSTATUS | Lookup | Set field = fldListRecord. Lookup (fieldName, index) |
| | | Map | Call fldListRecord. Map (baseFieldList, nameList) |
| | | MapName | Call fldListRecod. MapName (baseFieldList, nameList, mapList) |
| Merge | Merge (npNameFieldlist As NPFIELDLIST, npDataFieldlist As NPFIELDLIST, ByVal npMergeFlags As NPFLAGS) As NPSTATUS | Merge | Call fldListRecord. Merge (nameFieldList, dataFieldList, mergeFlags) |

*continued*

| *NPFieldlist Class NP Method* | *NP Syntax* | *LCFieldlist Class LC Method* | *LC Syntax* |
|---|---|---|---|
| MergeVirtual | MergeVirtual (npNameFieldlist As NPFIELDLIST, npDataFieldlist As NPFIELDLIST, ByVal npMergeFlags As NPFLAGS, ByVal npVirtualCode As NPINT, ByVal npVirtualFieldlist As Variant) As NPSTATUG | MergeVirtual | Call fldList. MergeVirtual (nameFieldList, dataFieldList, MergeFlags, virtualCode, virtualFieldList) |
| Remove | Remove (ByVal npIndex As NPINT) As NPSTATUS | Remove | Call fldLstRecord. Remove (index) |
| Replace | Replace (ByVal npIndex As NPINT, npFieldName As Variant, ByVal npDataType As NPTYPE, npNewField As Variant) As NPSTATUS | Replace | Call fldLstRecord. Replace (index, fieldName, dataType) |
| SetName | SetName (ByVal npIndex As NPINT, npFieldName As Variant) As NPSTATUS | SetName | Call fldLstRecord. SetName (index, fieldName) |

## NPFloat Class

| *NPFloat Class NP Method* | *NP Syntax* | *LC Method* | *LC Syntax* |
|---|---|---|---|
| FromCurrency | FromCurrency (npThisFloat As NPFOAT, npSrcCurrency As NPCURRENCY) As NPSTATUS | | |
| FromNumeric | FromNumeric (npThisFloat As NPFOAT, npSrcNumeric As NPNUMERIC) As NPSTATUS | | |
| FromStream | FromStream (npThisFloat As NPFOAT, npSrcStream As NPSTREAM) As NPSTATUS | | |
| FromStream | FromStream (npThisInt As NPINT, npSrcStream As NPSTREAM) As NPSTATUS | | |

## NPNumeric Class and LCNumeric Class

| *NPNumeric Class NP Method* | *NP Syntax* | *LCNumeric Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NPNumeric constructor | New () As NPNUMERIC | New | Dim variableName As New LCNumeric (long1, long2) |
| Add | Add (npNumeric1 As NPNUMERIC, npNumeric2 As NPNUMERIC) As NPSTATUS | Add | Call numericTotal. Add (numeric1, numeric2) |
| Clear | Clear () | | |
| Compare | Compare (npBaseNumeric As NPNUMERIC, npResult As NPINT) As NPSTATUS | Compare | result = numeric First. Compare (numericSecond) |
| Copy | Copy (npSrcNumeric As NPNUMERIC) | Copy | Set newNumeric = srcFirst. Copy |
| Create | Create (ByVal npPrecision As NPINT, ByVal npScale As NPINT) As NPSTATUS | | |
| FromFloat | FromFloat (ByVal npSrcFloat As NPFLOAT) As NPSTATUS | | |
| FromStream | FromStream (npSrcStream As NPSTREAM) As NPSTATUS | | |
| GetPrecision | GetPrecision () As NPINT | | |
| GetScale | GetScale () As NPINT | | |
| Subtract | Subtract (npNumeric1 As NPNUMERIC, npNumeric2 As NPNUMERIC) As NPSTATUS | Subtract | Call numericThird. Subtract (numericFirst, numericSecond) |

## NPStream Class and LCStream Class

| *NPStream Class NP Method* | *NP Syntax* | *LCStream Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NPStream constructor | New () As NPSTREAM | New | Dim variableName As New LCStream (maxLength, flags, format) |
| | | Append | Call newStream. Append (stream1, stream2) |
| Clear | Clear () | Clear | lcStream. Clear |
| Create | Create (ByVal npMaxLength As NPINT, ByVal npStreamFlags As NPFLAGS, ByVal npFormat As NPINT, As NPSTATUS | | |
| DataAlloc | DataAlloc (ByVal npStreamFormat As NPINT, ByVal npRequestLength As NPINT, npReserved1 As Variant, npLength As NPINT) As NPSTATUS | | |
| DataCompare | DataCompare (npBaseStream As NPSTREAM, npResult As NPINT) As NPSTATUS | Compare | result = thisStream. Compare (baseStream) |
| DataConvert | DataConvert (npSrcStream As NPSTREAM, npConvertFlags As NPFLAGS, ByVal npStreamFormat As NPINT, npReserved1 As Variant, npLength As NPINT) As NPSTATUS | Convert | Call newStream.Convert (srcStream, flags, format) |
| DataCopy | DataCopy (npSrcStream As NPSTREAM) As NPSTATUS | Copy | Set newStream = lcStream. Copy |
| DataExtract | DataExtract (npSrcStream As NPSTREAM, ByVal npOffset As NPINT, ByVal npLength As NPINT) As NPSTATUS | Extract | Call lcStream. Extract (srcStream, offset, length) |

*continued*

| *NPStream Class NP Method* | *NP Syntax* | *LCStream Class LC Method* | *LC Syntax* |
|---|---|---|---|
| DataFetch | DataFetch (ByVal npOffset As NPINT, ByVal npLength As NPINT, npReserved1 As Variant, npLength As NPINT) As NPSTATUS | | |
| DataFree | DataFree () | | |
| DataGetBuffer | DataGetBuffer (npStreamFormat As NPINT, npReserved1 As Variant) As NPINT | | |
| DataGetFlags | DataGetFlags () As NPFLAGS | | |
| DataGetInfo | DataGetInfo (npMaxLength As NPINT, npStreamFlags As NPFLAGS, npStreamFormat As NPINT, npReserved1 As Variant, npLength As NPINT) As NPSTATUS | | |
| DataGetLength | DataGetLength () As NPINT | | |
| DataMerge | DataMerge (npStream1 As NPSTREAM, ByVal npOffset As NPINT, npStream2 As NPSTREAM) As NPSTATUS | Merge | Call lcStream. Merge (stream1, offset, stream2) |
| DataRealloc | DataRealloc (ByVal npNewLength As NPINT, npReserved1 As Variant) As NPSTATUS | | |
| DataSet | DataSet (ByVal npStreamFormat As NPINT, npReserved1 As Variant, ByVal npLength As NPINT) As NPSTATUS | | |
| DataSetFlags | DataSetFlags (ByVal npStreamFlags As NPFLAGS) As NPSTATUS | | |
| DataTrim | DataTrim () As NPSTATUS | Trim | lcStream. Trim |
| FromCurrency | FromCurrency (npSrcCurrency As NPCURRENCY, ByVal npStreamFormat As NPINT) As NPSTATUS | | |

*continued*

| *NPStream Class NP Method* | *NP Syntax* | *LCStream Class LC Method* | *LC Syntax* |
|---|---|---|---|
| FromDatetime | FromDatetime (npSrcDatetime As NPDATETIME, ByVal npStreamFormat As NPINT) As NPSTATUS | | |
| FromFloat | FromFloat (ByVal npSrcFloat As NPFLOAT, ByVal npStreamFormat As NPINT) As NPSTATUS | | |
| FromInt | FromInt (ByVal npSrcInt As NPINT, ByVal npStreamFormat As NPINT) As NPSTATUS | | |
| FromNumeric | FromNumeric (npSrcNumeric As NPNUMERIC, ByVal npStreamFormat As NPINT) As NPSTATUS | | |
| | | ResetFormat | Call lcStream. ResetFormat (format) |
| | | SetFormat | Call lcStream. setFormat (format) |

## Stream List Data Class

| *Stream List Data Class NP Method* | *NP Syntax* | *Stream List Data Class LC Method* | *LC Syntax* |
|---|---|---|---|
| DatetimeList GetCount | DatetimeListGetCount (npValueCount As NPINT, npRangeCount As NPINT) | | |
| DatetimeList GetFirst | DatetimeListGetFirst (npDestDatetime As NPDATETIME) As NPSTATUS | | |
| DatetimeList GetLength | DatetimeListGetLength () As NPINT | | |

*continued*

| *Stream List Data Class NP Method* | *NP Syntax* | *Stream List Data Class LC Method* | *LC Syntax* |
|---|---|---|---|
| DatetimeList GetRange | DatetimeListGetRange (ByVal npIndex As NPINT, npDestDatetime1 As NPDATETIME, npDestDatetime2 As NPDATETIME) As NPSTATUS | DatetimeList GetRange | Call lcStream. DatetimeList GetRange (index, startDatetime, endDatetime) |
| DatetimeList GetValue | DatetimeListGetValue (ByVal npIndex As NPINT, npDestDatetime As NPDATETIME) As NPSTATUS | DatetimeList GetValue | Call lcStream. Datetime ListGetValue (index, datetime) |
| | | DatetimeList InsertRange | Call datetimeNew. DatetimeListInsertRange (index, startDatetime, endDatetime) |
| | | DatetimeList InsertValue | Call lcStream. DatetimeListInsertValue (index, datetime) |
| | | DatetimeList RemoveRange | Call lcStream. Datetime ListRemoveRange (index) |
| | | DatetimeList RemoveValue | Call lcStream. Datetime ListRemoveValue (index) |
| NumberList GetCount | NumberListGetCount (npValueCount As NPINT, npRangeCount As NPINT) | | |
| NumberList GetFirst | NumberListGetFirst (npDestFloat As NPFLOAT) As NPSTATUS | | |
| NumberList GetLength | NumberListGet Length () As NPINT | | |
| NumberList GetRange | NumberListGetRange (ByVal npIndex As NPINT, npDestFloat1 As NPFLOAT, npDestFloat2 As NPFLOAT) As NPSTATUS | NumberList GetRange | Call lcStream. NumberListGetRange (index, startNumber, endNumber) |

*continued*

| *Stream List Data Class NP Method* | *NP Syntax* | *Stream List Data Class LC Method* | *LC Syntax* |
|---|---|---|---|
| NumberList GetValue | NumberListGetValue (ByVal npIndex As NPINT, npDestFloat As NPFLOAT) As NPSTATUS | NumberList GetValue | Call lcStream. NumberListGetValue (index, number) |
| | | NumberList InsertRange | Call lcStream. Number ListInsertRange (index, startNumber, endNumber) |
| | | NumberList InsertValue | Call lcStream. NumberListInsert Value (index, number) |
| | | NumberList RemoveRange | Call lcStream. NumberListRemove Range (index) |
| | | NumberList RemoveValue | Call lcStream. Number ListRemoveValue (index) |
| TextListFetch | TextListFetch (ByVal npIndex As NPINT, ByVal npStreamFormat As NPINT, npDestStream As NPSTREAM, npReserved1 As Variant, npLength As NPINT) As NPSTATUS | TextListFetch | Call lcStream. TextListFetch (index, format, stream) |
| TextList GetCount | TextListGetCount () As NPINT | | |
| TextList GetLength | TextListGetLength (ByVal npIndex As NPINT) As NPINT | | |
| | | TextListInsert | Call lcStream. TextList Insert (index, stream) |
| | | TextList Remove | Call lcStream. Text ListRemove (index) |

# Appendix A
# Lotus Enterprise Integrator and Data

This appendix provides information about LEI data types and data mapping.

## Notes and Relational Databases

LEI can pass data between data sources, including, in some cases, Lotus Notes. It can be helpful in these cases to understand the match-up between Notes and database objects. The metadata object of an LEI Activity is the data table of a relational database or a Notes form. The following table summarizes LEI terminology in relation to Notes terms and traditional relational database vocabulary.

| *Type* | *Relational DB* | *Notes* |
|---|---|---|
| Database | Database | Database (.nsf file) |
| Metadata | Table | Form or View |
| Field | Column | Field or Column |
| Record | Row | Document or Row |
| Statement | SQL query | Selection formula |

## LEI Data Types

LEI uses the following data types internally when transferring and manipulating data. LEI converts an original data type to the closest matching LEI type and retains information about the original type. That information makes it possible to convert back to the original when appropriate.

| *Data Type* | *Parameters* | |
|---|---|---|
| INT | Description | 4-byte signed integer |
| | Maximum value | 2,147,483,647 |
| | Minimum value | -2,147,483,647 |
| | Precision | 9 digits |
| FLOAT | Description | 8-byte floating point value, corresponding to a C double |
| | Maximum positive value | 1.#J |
| | Minimum positive value | 0 |
| | Precision | 15 digits |
| CURRENCY | Description | 8-byte signed integer value: four fixed decimal places |
| | Maximum value | 922,337,203,685,478 |
| | Minimum value | -922,337,203,685,478 |
| | Accuracy | 0 |
| | Precision | 19 digits |
| NUMERIC | Description | High-precision packed numeric value |
| | Maximum positive value | 9.99…99e126 |
| | Minimum positive value | 0 |
| | Maximum precision | 88 digits |
| | Maximum scale | 127 |
| | Minimum scale | -127 |
| DATETIME | Description | 8-byte datetime value with timezone |
| | Maximum value | 12/31/32767 |
| | Minimum value | 1/1/1 |
| | Accuracy | 0.01 seconds |
| TEXT | Description | character-set specific text stream |
| | Properties | Fixed/Variable length, Case sensitivity |
| | Maximum length | 4 Gb |

*continued*

| *Data Type* | *Parameters* | |
|---|---|---|
| BINARY | Description | binary stream with optional binary format information |
| | Available formats | BLOB (unformatted), Composite (Notes Rich Text), Text List (multi-value Text), Number List (multi-value Number), Datetime List (multi-value Datetime), |
| | Properties | Fixed/Variable length, Case sensitivity |
| | Maximum length | 4 Gb |

## The Meaning of NULL

LEI regards a NULL as an undefined value, one that cannot be compared to other values. It is not equivalent to an empty string or to an empty value such as zero. A replication will occur, for example, if LEI finds a NULL in a database matching an empty value in the replica. In such cases, use SQL syntax to convert NULLs to values that are equivalent across databases.

LEI evaluates the absence of a field in a Notes database as a NULL.

## Year 2000 Solution

LEI uses the Pivot Date solution to the year 2000 date problem.

How LEI interprets two-digit dates is dependenat on the Century boundary setting in the lei.ini file. The CenturyBoundary variable can be set to any number between 0-101.

If the year is less than the boundary number, then the century 2000 will be used. If greater than or equal to the boundary, the century is 1900. If the boundary number is 101, then LEI uses the current century. The default boundary is 50. Zero always means always 1900 and 100 means always 2000.

Examples are shown below.

| | *How 2-digit years are translated* | | |
|---|---|---|---|
| CenturyBoundary = 0 | 00 = 1900 | 55 = 1955 | 99 = 1999 |
| CenturyBoundary = 55 | 00 = 2000 | 55 = 1955 | 99 = 1999 |
| CenturyBoundary = 60 | 00 = 2000 | 55 = 2055 | 99 = 1999 |
| CenturyBoundary = 100 | 00 = 2000 | 55 = 2055 | 99 = 2099 |

# Appendix B
# Common Gateway Interface (CGI) for LEI

This appendix provides information about configuring LEI to work with Common Gateway Interface (CGI).

## Using CGI for LEI

LEI ships with a Common Gateway Interface that allows it to be integrated into enterprise Internet/Intranet Hypertext Transfer Protocol (HTTP) servers for Web applications. Using the LEI CGI provides Web clients controlled access to any LEI supported database. The interface acts much like the command line interface, LEIACT. When the CGI program is launched from a Web server, all of the fields of the corresponding HTML form become extended properties of the activity.

See the LCSession class description in the *LotusScript Extension for Domino Connectors* for an explanation of extended properties.

Do not use spaces in activity names. If there are spaces in an activity name, you should replace them with a plus sign (+) in the command line syntax.

The LEI CGI program may need to be in the CG bin directory of your HTTP server. In addition, the LEI program directory will need to be in your system path. Check your Web server documentation for more information on executing CGI programs.

After creating an HTML form, use the syntax for the POST command as follows:

- For NT:
  ```
  <FORM METHOD="POST" ACTION="LEICGI.EXE?ActivityName">
  ```
- For AIX:
  ```
  <FORM METHOD="POST" ACTION="leicgi?ActivityName">
  ```
- For Solaris:
  ```
  <FORM METHOD="POST" ACTION="leicgi?ActivityName">
  ```

# Appendix C
# Character Sets

This appendix provides information about LEI and character sets, including character set translation, character sort order, and a list of the character sets supported by LEI.

## International Character Set Translation

Not all computer systems treat text data the same. Computers around the world have different character sets and sending text data from one country to another may not yield expected results. As well, different database products treat this problem differently.

LEI provides extensive support for international character sets. LEI supports over 200 different character set mappings, including US English, Asian, Far East, and Western and Eastern European languages, as well as multi-lingual characters sets such as LMBCS and Unicode. LEI provides support for translating characters from one set to another.

All text data within LEI maintains a reference to its own character set. This reference is determined automatically by the connector when the data is fetched from a database. If the data is inserted into a database with a different character set, LEI automatically translates the text data just prior to storing it. This process maintains maximum performance by eliminating any unnecessary translation.

Performance may be further enhanced through LEI's three levels of translation support:

- Full text data translation

  LEI will always translate between different character sets.
- LMBCS only

  Notes text data only is translated. This is appropriate when all non-Notes text data is in compatible character sets.
- No translation

  For maximum performance and when all text data is in compatible character sets, translation may be disabled completely.

Regardless of the setting, Unicode text is always translated.

The text translation setting is available for each LEI server, allowing different servers to have different levels of support. The translation support level is set during the installation of an LEI server. It may also be viewed and edited in the server configuration document in the LEI Administrator.

## Forcing LEI to use a Character Set as Native

Lotus Enterprise Integrator (LEI) determines the native character set on the local machine during startup and uses this setting whenever the native character set is required. LEI can be forced to use a different character set as native by setting the NativeText variable in the lei.ini file to the suffix of a LEI text stream format.

LCSTREAMFMT_IBMCP932 indicates IBM code page 932. To force LEI to use that character set as native, add the following line to lei.ini:

```
NativeText=IBMCP932
```

**Note** The Supported Character Sets table appears later in this appendix. You can specify the native text character set in the lei.ini by simply removing the LCSTREAMFMT preface from the IBM*name* portion of the Stream Format Constant name listed in that table.

In all cases, this character set will be assumed to be the native format.

The NativeText variable can be used in conjunction with any Character Set Translation setting. The Character Set Translation Disabled setting causes this character set to be used in all connector interactions. For example, if you're on a CP932 machine, and you access CP819 data with the NativeText variable set to IBMCP932, the character set is considered to be CP932 without translation enabled. If converted to another character set, it is converted as if it were CP932.

In LMBCS-only mode, the system only operates with three character sets:

- Unicode = Unicode
- LMBCS = LMBCS
- All else = Native codepage

This option changes what native text is on the machine. For example, if you force native text to be CP819, then all native text conversion maps to and from CP819. You still have Unicode and LMBCS as in the above example, but everything else is considered CP819.

This feature is important for Thai systems, where the native character set generally cannot be retrieved well. The Thai character set, IBMCP874, can be set and then with the translation level set to LMBCS-only or disabled, all databases that LEI interacts with (except Notes if LMBCS only translation is selected) will be considered as CP874.

## Character Sort Order

LEI cannot control the sort order (collation) specified in the databases that it accesses. If the sort orders used in two databases are not the same, mismatches can arise during Replication Activities, causing excessive insert/delete operations and thereby affecting performance. The use of the Order MetaConnector for databases which sort differently is designed to solve sort order differences. See Chapter 16, "MetaConnectors" for more information. Replication also provides an option for case-insensitive string comparison when excessive insert/delete pairs occur during replication between databases that use case-insensitive sorting. See Chapter 26, "Replication Activity" for more information about this feature.

## List of Supported Character Sets

The list is given as text stream format constants. Any of these values may be used as a stream format for a text stream when creating activity scripts using the LEI LotusScript Extensions. To indicate the character set on the local machine, use the constant LCSTREAMFMT_NATIVE. When providing a character set on the Administrator, use the stream format constant with the prefix "LCSTREAMFMT_" removed.

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_LICS | Lotus International Character Set |
| LCSTREAMFMT_IBMCP851 | MS-DOS PC Greek (CP 851) |
| LCSTREAMFMT_IBMCP852 | MS-DOS PC Eastern European (CP 852) |
| LCSTREAMFMT_IBMCP853 | MS-DOS PC Turkish (CP 853) |
| LCSTREAMFMT_IBMCP857 | MS-DOS PC Turkish (CP 857) |
| LCSTREAMFMT_IBMCP862 | MS-DOS PC Hebrew (CP 862) |
| LCSTREAMFMT_IBMCP864 | MS-DOS PC Arabic (CP 864) |
| LCSTREAMFMT_IBMCP866 | MS-DOS PC Cyrillic Unicode (CP 866) |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_IBMCP437 | MS-DOS PC US (CP 437) |
| LCSTREAMFMT_IBMCP850 | MS-DOS PC Western European (CP 850) |
| LCSTREAMFMT_IBMCP855 | MS-DOS PC Cyrillic (CP 855) |
| LCSTREAMFMT_IBMCP860 | MS-DOS PC Portuguese (CP 860) |
| LCSTREAMFMT_IBMCP861 | MS-DOS PC Icelandic (CP 861) |
| LCSTREAMFMT_IBMCP863 | MS-DOS PC Canadian French (CP 863) |
| LCSTREAMFMT_IBMCP865 | MS-DOS PC Norwegian (CP 865) |
| LCSTREAMFMT_IBMCP869 | MS-DOS PC Greek (CP 869) |
| LCSTREAMFMT_IBMCP899 | IBM Code Page 899 (CP 899) |
| LCSTREAMFMT_IBMCP932 | MS-DOS PC Japanese Microsoft Shift-JIS (CP 932) |
| LCSTREAMFMT_IBMCP942 | MS-DOS PC Japanese Microsoft Shift-JIS (CP 942) |
| LCSTREAMFMT_IBMCP891 | MS-DOS PC Korean (CP 891) |
| LCSTREAMFMT_DECMCS | DEC Multinational Character Set |
| LCSTREAMFMT_EUC | Extended UNIX Code |
| LCSTREAMFMT_KS | MS-DOS Korean – KSC 5601 |
| LCSTREAMFMT_IBMCP949 | MS-DOS Korean (CP 949) |
| LCSTREAMFMT_TCA | TCA |
| LCSTREAMFMT_BIG5 | MS-DOS Taiwan (traditional) Chinese (BIG-5) |
| LCSTREAMFMT_IBMCP950 | MS-DOS Taiwan (traditional) Chinese (CP 950) |
| LCSTREAMFMT_GB | MS-DOS PRC (simplified) Chinese (GB 2312) |
| LCSTREAMFMT_IBMCP936 | MS-DOS PRC (simplified) Chinese (CP 936) |
| LCSTREAMFMT_NECESJIS | MS-DOS PC Japanese NEC Shift-JIS (CP 932) |
| LCSTREAMFMT_ISO646 | ASCII |
| LCSTREAMFMT_ASCII | ASCII |
| LCSTREAMFMT_ISO88591 | ISO Latin-1 US, Western European (ISO-8859-1) |
| LCSTREAMFMT_IBMCP819 | ISO Latin-1 US, Western European (CP 819) |
| LCSTREAMFMT_ISO88592 | ISO Latin-2 Eastern European (ISO-8859-2) |
| LCSTREAMFMT_IBMCP912 | ISO Latin-2 Eastern European (CP 912) |
| LCSTREAMFMT_ISO88593 | ISO Latin-3 Southern European (ISO-8859-3) |
| LCSTREAMFMT_ISO88594 | ISO Latin-4 Northern European (ISO-8859-4) |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_ISO88595 | ISO Cyrillic (ISO-8859-5) |
| LCSTREAMFMT_IBMCP915 | ISO Cyrillic (CP 915) |
| LCSTREAMFMT_ISO88596 | ISO Arabic (ISO-8859-6) |
| LCSTREAMFMT_IBMCP1008 | ISO Arabic (CP 1008) |
| LCSTREAMFMT_ISO88597 | ISO Greek (ISO-8859-7) |
| LCSTREAMFMT_IBMCP813 | ISO Greek (CP 813) |
| LCSTREAMFMT_ISO88598 | ISO Hebrew (ISO-8859-8) |
| LCSTREAMFMT_IBMCP916 | ISO Hebrew (CP 916) |
| LCSTREAMFMT_ISO88599 | ISO Latin-5 Southern European (ISO-8859-9) |
| LCSTREAMFMT_IBMCP920 | ISO Latin-5 Southern European (CP 920) |
| LCSTREAMFMT_HPROMAN | HP Roman (Laserjet) |
| LCSTREAMFMT_HPGREEK | HP Greek (Laserjet) |
| LCSTREAMFMT_HPTURKISH | HP Turkish (Laserjet) |
| LCSTREAMFMT_HPHEBREW | HP Hebrew (Laserjet) |
| LCSTREAMFMT_HPARABIC | HP Arabic (Laserjet) |
| LCSTREAMFMT_HPTHAI | HP Thai (Laserjet) |
| LCSTREAMFMT_HPJAPAN | HP Japanese (Laserjet) |
| LCSTREAMFMT_HPKANA | HP Kana (Laserjet) |
| LCSTREAMFMT_HPKOREA | HP Korean (Laserjet) |
| LCSTREAMFMT_HPPRC | HP Simplified Chinese (Laserjet) |
| LCSTREAMFMT_HPROC | HP Traditional Chinese (Laserjet) |
| LCSTREAMFMT_IBMCP37 | IBM EBCDIC US/Canadian English (CP 37) |
| LCSTREAMFMT_IBMCP28709 | IBM Code Page 28709 (CP28709) |
| LCSTREAMFMT_IBMCP273 | IBM EBCDIC German – Austrian (CP 273) |
| LCSTREAMFMT_IBMCP278 | IBM EBCDIC Finnish, Swedish (CP 278) |
| LCSTREAMFMT_IBMCP280 | IBM EBCDIC Italian (CP 280) |
| LCSTREAMFMT_IBMCP284 | IBM EBCDIC Spanish, Latin American (CP 284) |
| LCSTREAMFMT_IBMCP285 | IBM EBCDIC UK (CP 285) |
| LCSTREAMFMT_IBMCP290 | IBM EBCDIC Japanese (Katakana) (CP 290) |
| LCSTREAMFMT_IBMCP297 | IBM EBCDIC French (CP 297) |
| LCSTREAMFMT_IBMCP500 | IBM EBCDIC International (CP 500) |
| LCSTREAMFMT_IBMCP277 | IBM EBCDIC Danish, Norwegian (CP 277) |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_IBMCP1047 | IBM EBCDIC Latin-1 Open Systems (CP 1047) |
| LCSTREAMFMT_IBMCP1250 | Windows Eastern European (CP 1250) |
| LCSTREAMFMT_IBMCP1251 | Windows Cyrillic (CP 1251) |
| LCSTREAMFMT_IBMCP1252 | Windows ANSI (CP 1252) |
| LCSTREAMFMT_ANSI | ANSI |
| LCSTREAMFMT_IBMCP1253 | Windows Greek (CP 1253) |
| LCSTREAMFMT_IBMCP1254 | Windows Turkish (CP 1254) |
| LCSTREAMFMT_IBMCP1255 | Windows Hebrew (CP 1255) |
| LCSTREAMFMT_IBMCP1256 | Windows Arabic (CP 1256) |
| LCSTREAMFMT_IBMCP1257 | Windows Baltic (CP 1257) |
| LCSTREAMFMT_IBMCP1363 | Windows Korean (CP 1363) |
| LCSTREAMFMT_MACSCRIPT0 | Macintosh Western European (Script 0) |
| LCSTREAMFMT_MACSCRIPT1 | Macintosh Japanese (Script 1) |
| LCSTREAMFMT_MACSCRIPT6 | Macintosh Greek (Script 6) |
| LCSTREAMFMT_MACSCRIPT7 | Macintosh Cyrillic (Script 7) |
| LCSTREAMFMT_MACSCRIPT29 | Macintosh Central Europe (Script 29) |
| LCSTREAMFMT_MACSCRIPT0 TURKISH | Macintosh Roman variant – Turkish (Script 81) |
| LCSTREAMFMT_THAI | MS Thai Windows |
| LCSTREAMFMT_IBMCP874 | MS-DOS PC Thai (CP 874) |
| LCSTREAMFMT_ISO885911 | ISO Thai (ISO-8859-11) |
| LCSTREAMFMT_TIS620 | Thai Industrial Standard (TIS620-2529) |
| LCSTREAMFMT_UNICODE | Unicode (ISO 10646) |
| LCSTREAMFMT_IBMCP1200 | Unicode (IBM CP 1200) |
| LCSTREAMFMT_ISO10646 | Unicode (ISO 10646) |
| LCSTREAMFMT_UTF7 | Unicode Transformation Formats 7 |
| LCSTREAMFMT_UTF8 | Unicode Transformation Formats 8 |
| LCSTREAMFMT_LMBCS | Lotus MultiByte Character Set (LMBCS) |
| LCSTREAMFMT_DECNRCUK | DEC National Replacement Char – UK |
| LCSTREAMFMT_DECNRCDUTCH | DEC Nat'l Replacement Char – Dutch |
| LCSTREAMFMT_DECNRCFINNISH | DEC Nat'l Replacement Char – Finnish |
| LCSTREAMFMT_DECNRCFRENCH | DEC Nat'l Replacement Char – French |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_DECNRCFRENCH CANADIAN | DEC Nat'l Replacement Char – French Canadian |
| LCSTREAMFMT_DECNRC GERMAN | DEC Nat'l Replacement Char – German |
| LCSTREAMFMT_DECNRCITALIAN | DEC Nat'l Replacement Char – Italian |
| LCSTREAMFMT_DECNRC NORWEGIANDANISH | DEC Nat'l Replacement Char – Norwegian Danish |
| LCSTREAMFMT_DECNRC PORTUGUESE | DEC Nat'l Replacement Char – Portuguese |
| LCSTREAMFMT_DECNRCSPANISH | DEC Nat'l Replacement Char – Spanish |
| LCSTREAMFMT_DECNRC SWEDISH | DEC Nat'l Replacement Char – Swedish |
| LCSTREAMFMT_DECNRCSWISS | DEC Nat'l Replacement Char – Swiss |
| LCSTREAMFMT_T61 | Teletex T.61 |
| LCSTREAMFMT_T50 | Teletex T.50 |
| LCSTREAMFMT_ASN1 | ANSI Standard Notation (ASN.1) |
| LCSTREAMFMT_IBMCP856 | MS-DOS PC Hebrew (CP 85) |
| LCSTREAMFMT_IBMCP1004 | MS-DOS PC Desktop Publishing (CP 1004) |
| LCSTREAMFMT_IBMCP1002 | IBM EBCDIC DCF (CP 1002) |
| LCSTREAMFMT_IBMCP1003 | IBM EBCDIC US Text Subset (CP 1003) |
| LCSTREAMFMT_IBMCP1025 | IBM EBCDIC Cyrillic (CP 1025) |
| LCSTREAMFMT_IBMCP1026 | IBM EBCDIC Turkish (CP 1026) |
| LCSTREAMFMT_IBMCP1028 | IBM EBCDIC Hebrew Publishing (CP 1028) |
| LCSTREAMFMT_IBMCP256 | IBM EBCDIC International #1 (CP 256) |
| LCSTREAMFMT_IBMCP259 | IBM EBCDIC Symbols Set 7 (CP 259) |
| LCSTREAMFMT_IBMCP274 | IBM EBCDIC Belgian (CP 274) |
| LCSTREAMFMT_IBMCP275 | IBM EBCDIC Brazilian (CP 275) |
| LCSTREAMFMT_IBMCP281 | IBM EBCDIC Japanese (Latin) (CP 281) |
| LCSTREAMFMT_IBMCP282 | IBM EBCDIC Portuguese (CP 282) |
| LCSTREAMFMT_IBMCP361 | IBM EBCDIC International #5 (CP 361) |
| LCSTREAMFMT_IBMCP382 | IBM EBCDIC Austrian, German, Switzerland (CP 382) |
| LCSTREAMFMT_IBMCP383 | IBM EBCDIC Belgian (CP 383) |
| LCSTREAMFMT_IBMCP384 | IBM EBCDIC Brazilian (CP 384) |
| LCSTREAMFMT_IBMCP385 | IBM EBCDIC Canadian (French) (CP 385) |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_IBMCP386 | IBM EBCDIC Danish, Norwegian (CP 386) |
| LCSTREAMFMT_IBMCP387 | IBM EBCDIC Finnish, Swedish (CP 387) |
| LCSTREAMFMT_IBMCP388 | IBM EBCDIC French, Swiss (CP 388) |
| LCSTREAMFMT_IBMCP389 | IBM EBCDIC Italian, Swiss (CP 389) |
| LCSTREAMFMT_IBMCP390 | IBM EBCDIC Japanese (Latin) (CP 390) |
| LCSTREAMFMT_IBMCP391 | IBM EBCDIC Portuguese (CP 391) |
| LCSTREAMFMT_IBMCP392 | IBM EBCDIC Spanish, Philippines (CP 392) |
| LCSTREAMFMT_IBMCP393 | IBM EBCDIC Latin American (Spanish Speaking) (CP 393) |
| LCSTREAMFMT_IBMCP394 | IBM EBCDIC UK, Australian, Hong Kong, Ireland, New Zealand (CP 394) |
| LCSTREAMFMT_IBMCP395 | IBM EBCDIC US, Canadian (English) (CP 395) |
| LCSTREAMFMT_IBMCP423 | IBM EBCDIC Greek 183 (CP 423) |
| LCSTREAMFMT_IBMCP424 | IBM EBCDIC Hebrew (CP 424) |
| LCSTREAMFMT_IBMCP803 | IBM EBCDIC Hebrew Character Set A (CP 803) |
| LCSTREAMFMT_IBMCP870 | IBM EBCDIC Eastern Europe (CP 870) |
| LCSTREAMFMT_IBMCP871 | IBM EBCDIC Icelandic (CP 871) |
| LCSTREAMFMT_IBMCP875 | IBM EBCDIC Greek (CP 875) |
| LCSTREAMFMT_IBMCP880 | IBM EBCDIC Cyrillic (CP 880) |
| LCSTREAMFMT_IBMCP905 | IBM EBCDIC Turkish (CP 905) |
| LCSTREAMFMT_IBMCP948 | IBM Extended Taiwanese (CP 948) |
| LCSTREAMFMT_IBMCP938 | IBM Taiwanese (CP 938) |
| LCSTREAMFMT_IBMCP1381 | IBM GBK = GB + Hanzi (CP 1381) |
| LCSTREAMFMT_IBMCP1383 | IBM Traditional Chinese (CP 1383) |
| LCSTREAMFMT_IBMCP1386 | IBM Traditional Chinese (CP 1386) |
| LCSTREAMFMT_EACC | East Asian Character Code Set (ANSI Z39.64-1989) |
| LCSTREAMFMT_JIS | Japanese Information Standard 0201 (JIS 201) |
| LCSTREAMFMT_CCCII | Chinese Character Code for Information Interchange (Taiwan) |
| LCSTREAMFMT_XEROXCJK | Xerox CJK |
| LCSTREAMFMT_IBMCP944 | IBM Extended Korean (CP 944) |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_IBMCP934 | IBM Korean (CP 934) |
| LCSTREAMFMT_IBMCP737 | MS-DOS PC Greek (CP 737) |
| LCSTREAMFMT_IBMCP775 | MS-DOS PC Baltic (CP 775) |
| LCSTREAMFMT_ISO6937 | Latin chars (non-spacing accents) similar to T.61 |
| LCSTREAMFMT_BASE64 | Content-Transfer-Encoding |
| LCSTREAMFMT_JIS2 | Japanese Information Standard 0208 (JIS 208) |
| LCSTREAMFMT_EUCJ | Extended UNIX Code – Japanese |
| LCSTREAMFMT_EUCT | Extended UNIX Code – Taiwanese |
| LCSTREAMFMT_EUCK | Extended UNIX Code – Korean |
| LCSTREAMFMT_ISOKR | ISO-2022-KR switching: treated as EUCK |
| LCSTREAMFMT_EUCC | Extended UNIX Code – Chinese |
| LCSTREAMFMT_IBMCP921 | Replacement for Lithuanian (CP 921) |
| LCSTREAMFMT_IBMCP922 | Russian (CP 922) |
| LCSTREAMFMT_KOI8 | Cyrillic Internet Support |
| LCSTREAMFMT_IBMCP720 | IBM Code Page 720 (CP 720) |
| LCSTREAMFMT_IBMCP1258 | Windows Vietnamese (CP 1258) |
| LCSTREAMFMT_ISO885910 | ISO Latin-6 (ISO-8859-10) |
| LCSTREAMFMT_JP1TEXT | OSI/JIS X 5003-1987 X.400 Japanese ISP |
| LCSTREAMFMT_VIQRI | Vietnamese Quoted Readable |
| LCSTREAMFMT_VISCII | Vietnamese VISCII 1.1 (VICSII) |
| LCSTREAMFMT_VISCII1 | TCVN Vietnamese Orthographic (VCSII-1) |
| LCSTREAMFMT_VISCII2 | TCVN Vietnamese Graphic (VCSII-2)*/ |
| LCSTREAMFMT_IBMCP838 | IBM EBCDIC SBCS Thai (CP 838) |
| LCSTREAMFMT_IBMCP9030 | IBM EBCDIC SBCS Thai (CP 9030) |
| LCSTREAMFMT_IBMCP833 | IBM EBCDIC SBCS Korean – extended (CP 833) |
| LCSTREAMFMT_IBMCP836 | IBM EBCDIC SBCS PRC (simplified) Chinese (CP 836) |
| LCSTREAMFMT_IBMCP1027 | IBM EBCDIC SBCS Japanese Latin – extended (CP 1027) |
| LCSTREAMFMT_IBMCP420 | IBM EBCDIC Arabic (CP 420) |
| LCSTREAMFMT_IBMCP918 | IBM EBCDIC Code Page 918 (CP 918) |
| LCSTREAMFMT_IBMCP1097 | IBM EBCDIC Code Page 1097 (CP 1097) |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_IBMCP1112 | IBM EBCDIC Code Page 1112 (CP 1112) |
| LCSTREAMFMT_IBMCP1122 | IBM EBCDIC Code Page 1122 (CP 1122) |
| LCSTREAMFMT_IBMCP1123 | IBM EBCDIC Code Page 1123 (CP 1123) |
| LCSTREAMFMT_IBMCP1129 | IBM EBCDIC Code Page 1129 (CP 1129) |
| LCSTREAMFMT_IBMCP1130 | IBM EBCDIC Code Page 1130 (CP 1130) |
| LCSTREAMFMT_IBMCP1132 | IBM EBCDIC Code Page 1132 (CP 1132) |
| LCSTREAMFMT_IBMCP1133 | IBM EBCDIC Code Page 1133 (CP 1133) |
| LCSTREAMFMT_IBMCP806 | ISO Devnagiri (CP 806) |
| LCSTREAMFMT_IBMCP1137 | IBM EBCDIC Devnagiri (CP 1137) |
| LCSTREAMFMT_VISCII3 | TCVN3 Vietnamese (VCSII-3) |
| LCSTREAMFMT_TCVN3 | TCVN3 Vietnamese (VCSII-3) |
| LCSTREAMFMT_IBMCP930 | IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 930) |
| LCSTREAMFMT_IBMCP933 | IBM EBCDIC EUC Korean Mixed (CP 933) |
| LCSTREAMFMT_IBMCP935 | IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 935) |
| LCSTREAMFMT_IBMCP937 | IBM EBCDIC EUC Taiwan (traditional) Chinese Mixed (CP 937) |
| LCSTREAMFMT_IBMCP939 | IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 939) |
| LCSTREAMFMT_IBMCP931 | IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 931) |
| LCSTREAMFMT_IBMCP1388 | IBM EBCDIC EUC PRC (simplified) Chinese Mixed (CP 1388) |
| LCSTREAMFMT_IBMCP5026 | IBM EBCDIC EUC Japanese Katakana Kanji Mixed (CP 5026) |
| LCSTREAMFMT_IBMCP5035 | IBM EBCDIC EUC Japanese Latin Kanji Mixed (CP 5035) |
| LCSTREAMFMT_IBMCP300 | IBM EBCDIC DBCS Japanese (CP 300) |
| LCSTREAMFMT_IBMCP834 | IBM EBCDIC DBCS Korean (CP 834) |
| LCSTREAMFMT_IBMCP835 | IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 835) |
| LCSTREAMFMT_IBMCP837 | IBM EBCDIC DBCS PRC (simplified) Chinese (CP 837) |
| LCSTREAMFMT_IBMCP930X | IBM EBCDIC DBCS Japanese (CP 930X) |

*continued*

| *Stream Format Constant* | *Description* |
|---|---|
| LCSTREAMFMT_IBMCP933X | IBM EBCDIC DBCS Korean (CP 933X) |
| LCSTREAMFMT_IBMCP935X | IBM EBCDIC DBCS PRC (simplified) Chinese (CP 935X) |
| LCSTREAMFMT_IBMCP937X | IBM EBCDIC DBCS Taiwan (traditional) Chinese (CP 937X) |
| LCSTREAMFMT_IBMCP939X | IBM EBCDIC DBCS Japanese (CP 939X) |
| LCSTREAMFMT_IBMCP931X | IBM EBCDIC DBCS PRC (simplified) Chinese (CP 931X) |

## Language Environment Variables for UNIX Platforms

UNIX systems use the LANG environment variable to set the locale (language/country/codepage) for applications to use. The applications don’t have to respect the value, but it’s there if they want/need to use it. LEI uses cc:STR, which takes that value and maps it to the cc:STR codepage/country code/language code values according to preset tables.

See your platform manuals for appropriate LANG settings.

# Appendix D
# Error Messages and Troubleshooting

This appendix provides information for troubleshooting LEI.

## Installation Troubleshooting

See the *LEI Domino Connectivity and Installation Guide* for information about troubleshooting the LEI installation.

If Notes, Domino, or LEI reports the following message, it may be because you have upgraded Domino or Notes to release 4.6.n and above after you installed LEI.

The procedure entry point LCStreamCase / LCFieldlistGetDefinition could not be located in library LCAPI.

Domino or Notes replaces some of the LEI program files with older versions and requires that you reinstall LEI.

**Prior to the next Release of Domino (and DECS 5.0.5), if you reinstall or upgrade Domino with DECS after installing LEI 3.1, you must reinstall or restore LEI.**

For more information, see the following two Web sites:

**`www.lotus.com/dominoei`**

**`www.lotus.com/developers`**

## List of LEI Error Messages

This section provides brief explanations of LEI’s error messages.

## Basic Errors

**LCFAIL_MEMORY:** Unable to allocate memory.

A memory allocation request failed. Either reduce memory requirements or free up system resources. Common causes for this error include extremely large fieldlist record count or fixed stream length.

**LCFAIL_UNAVAILABLE:** Requested functionality is not available.

A request cannot be satisfied. This is most commonly returned from a Connection which does not support a particular function or operation, for example a read-only Connection would return. This in response to an Insert operation.

**LCFAIL_END_OF_DATA:** The last data value has been retrieved.

This indicates the normal end of processing from any iterative operation, and should not normally be treated as an error condition. All iterative operations, such as Fetching from a result set or listing fields, will eventually return this status code indicating that there are no more results available.

**LCFAIL_INVALID_INDEX:** Cannot locate list element.

An index parameter provided is beyond the range of the object. For example, requesting the third record from a two-record fieldlist would return this error.

**LCFAIL_INVALID_LIST:** Invalid List direction.

An LCLIST enumeration parameter was not valid. One of the LCLIST_XXX constants must be provided for any List parameter.

**LCFAIL_INVALID_CONVERT:** Invalid conversion.

Conversion is not supported between the source and target types. For example, conversion between numbers and datetimes will result in this error.

**LCFAIL_INVALID_TEXT_LIST:** This operation requires a valid text list.

An LCTextList function requires lcThisTextList to be a stream with a valid text list format (format LCSTREAMFMT_TEXT_LIST and a length of at least 2) or a single text value (type LEITYPE_TEXT). Either one of these conditions was not satisfied or the data was not a valid text list.

**LCFAIL_INVALID_NUMBER_LIST:** This operation requires a valid number list.

An LCNumberList function requires lcThisNumberList to be a stream with a valid number list format (format LCSTREAMFMT_NUMBER_LIST and a length of at least 4). Either one of these conditions was not satisfied or the data was not a valid number list.

**LCFAIL_INVALID_DATETIME_LIST:** This operation requires a valid datetime list.

An LCDatetimeList function requires lcThisDatetimeList to be a stream with a valid datetime list format (format LCSTREAMFMT_DATETIME_LIST and a length of at least 4). Either one of these conditions was not satisfied or the data was not a valid datetime list.

**LCFAIL_ZERO_INDEX:** All index values are one-based – an index of zero is not valid.

An index parameter provided is zero, and all index values for LEI objects are one-based.

**LCFAIL_ZERO_COUNT:** This operation requires a non-zero count.

A count parameter provided is zero, which is not valid in the current context. Most, but not all, count parameters must be one or greater.

**LCFAIL_ZERO_OFFSET:** All offset values are one-based – an offset of zero is not valid.

An offset parameter provided is zero, and all offset values for LEI objects are one-based.

**LCFAIL_ZERO_FORMAT:** This operation requires a non-zero Stream Format.

A stream format parameter provided is zero, which is not valid in the current context. Most, but not all, stream format parameters must be a valid LCSTREAMFMT_XXX constant.

**LCFAIL_NULL_BUFFER:** A NULL buffer was provided when one was required.

A NULL pointer was provided for a memory buffer parameter, which is not valid in the current context. Some buffer parameters (usually input buffers) must be valid pointers.

**LCFAIL_NULL_RESULT:** A return parameter is required, but none was provided.

A NULL pointer was provided for an output parameter. Some output parameters (such as created objects) require valid pointers.

**LCFAIL_FIXED_LENGTH:** A fixed-length stream requires a non-zero length.

An LCSTREAM object with the flag LCSTREAMF_FIXED allocates a fixed-length buffer at stream creation time. The length is determined by the stream's maximum length property, which cannot be zero for a fixed-length stream.

**LCFAIL_INVALID_FLAGS:** The supplied flags are invalid, possibly due to a conflict.

A flags parameter contains invalid or conflicting flags. A few flags constants cannot be used together (for example, LCFIELDF_KEY_GT and LCFIELDF_KEY_LT).

**LCFAIL_TEXT_TRANSLATE:** Text translation failure.

Translation between text stream formats failed.

**LCFAIL_NULL_FIELDNAME:** A NULL field name was provided.

A NULL pointer was provided for a field name parameter. A valid pointer is required.

**LCFAIL_INVALID_FIELDLIST:** Invalid fieldlist.

An invalid LCFIELDLIST object instance/handle was used.

**LCFAIL_INVALID_CONNECT:** Invalid connect.

An invalid LCCONNECT object instance/handle was used.

**LCFAIL_EMPTY_FIELDLIST:** This operation cannot be performed on a fieldlist with no fields.

A fieldlist with no fields was used in an operation that is invalid on an empty fieldlist. Add one or more fields to the fieldlist and retry the operation.

**LCFAIL_NAME_FIELDLIST:** This operation cannot be performed on a name-only fieldlist.

A fieldlist allocated with a record count of zero contains field names only and no field data. Such a fieldlist cannot be used in the current context. Use a fieldlist allocated with a record count of one or greater.

**LCFAIL_FIELDLIST_REF:** Cannot alter fields in a multiply-referenced fieldlist.

A fieldlist created as a reference copy of another fieldlist shares field data space with the original fieldlist. Fields cannot be added to or removed from such a fieldlist as long as one copy still exists.

**LCFAIL_RECORD_INDEX:** An invalid fieldlist record index was encountered.

An lcRecordIndex parameter was invalid for the corresponding fieldlist. A record index must be no greater than the number of fields in the fieldlist.

**LCFAIL_RECORD_COUNT:** Request to transfer more records than allocated in fieldlist.

An lcRecordCount parameter was invalid for the corresponding fieldlist. A record count must be no greater than the number of fields in the fieldlist, less the record index.

**LCFAIL_LIST_SETUP:** Fieldlist iteration requires initial setup.

Iterating through fields in a fieldlist with LCFieldlistList requires that LCFieldlistListSetup be called first to prepare the fieldlist for iteration. When LCFieldlistList is called before LCFieldlistListSetup for a particular fieldlist instance, this error is returned.

**LCFAIL_NO_MERGE_DATA:** The data fieldlist in a merge cannot be a name-only fieldlist.

A fieldlist allocated with a record count of zero contains field names only and no field data. Such a fieldlist cannot be used as the lcDataFieldlist parameter for an LCFieldlistMerge or LCFieldlistMergeVirtual operation.

**LCFAIL_NO_RESULTSET:** This operation requires an active result set.

The requested operation cannot be performed against a Connection without an active result set. A result set must first be created in the Connection with LCConnectExecute, LCConnectSelect, or LCConnectCatalog.

**LCFAIL_NO_WRITEBACK:** This operation requires an active writeback result set.

The requested operation cannot be performed against a Connection without an active writeback context. A writeback result set (WRITEBACK property set to TRUE) must first be created in the Connection and a successful Fetch operation performed before a writeback Update or Remove operation is valid.

**LCFAIL_WRITEBACK_COUNT:** Writeback operation record counts must be one.

Writeback Updates and Removes operate on the current record in a Connection's result set. For these operations, the lcRecordCount parameter must be one.

**LCFAIL_STATIC_PROPERTY:** Static activity properties cannot be used with unnamed activities.

Activities initiated with LCActivityAlloc and no activity name exist outside the context of the LEI Administrator. Since static properties are stored within the activity document in the Administrator, they are not valid for unnamed activities.

**LCFAIL_LCXINIT:** LEI system has not been initialized.

The LEI API system must be initialized before calling any LEI functions. The system is initialized by command-line execution of an Activity or calling LCActivityAlloc. This error is returned when LEI API functions are called before the system is initialized.

**LCFAIL_ACTIVITY_NOT_INIT:** The Activity must be initialized before performing any operation.

The Activity context must be initialized before calling any LCActivity functions. The Activity context is initialized by command-line execution of an Activity or calling LCActivityAlloc. This error is returned when Activity functions are called before the Activity context is initialized.

**LCFAIL_ACTRUN_TIMEOUT:** Timeout while running synchronous activities.

When executing synchronous activities via LCActivityRunActivity and using a timeout, this error is returned if the timeout occurs before all synchronous activities and their children have completed.

**LCFAIL_NOT_CONNECTED:** This operation requires a connection to a Connector. This error is returned by LCConnect functions which require a successful call to LCConnectConnect. All functions which manipulate or retrieve data require a successful connection.

**LCFAIL_CONNECTED:** This operation cannot be performed with a valid Connector connection.

This error is returned by Connectors which require a particular operation to be performed before establishing a connection with LCConnectConnect.

**LCFAIL_EXTERNAL:** External error.

This error indicates that an error external to LEI has been produced. The specific external error code and error text are usually available to provide additional information.

**LCFAIL_ACTIVE_SUBCONNECTION**

The subConnection (Connection property) of a MetaConnection can only be set once. This error occurs when setting the property, but a valid setting has already been made.

## Basic Events (Non-Error)

**LCEVENT_EXTERNAL:** External event

This status code indicates that an event external to LEI has been produced. This is not an error situation, but rather an informational message. The specific external event code and event text are usually available to provide additional information.

## Extended Errors

**LCFAIL_INVALID_METADATA:** Metadata object <metadata> does not exist.

The metadata object to use is provided in the METADATA property. The current value for this property does not represent a valid metadata object in this Connection. Note that you can get this message when working with AS400 data and have not enabled the Non-Journaled Data option. In general, when working with AS400 data, you should enable the Non-Journaled Data option.

**LCFAIL_TYPE_MISMATCH:** Type mismatch for field <fieldname>; LEI: <type>, Connector: <type>.

When matching LEI fields to Connection fields, the basic type class must match. They must both be numbers, datetimes, or streams. This error is returned when there is a data type mismatch.

**LCFAIL_DUPLICATE:** Duplicate object <name>.

An attempt to create an object failed since an object of the same name already exists.

**LCFAIL_FIELD_COUNT_MISMATCH:** Field count mismatch; LEI: <count>, Connector: <count>.

When mapping fields, mismatched fields are only valid in certain circumstances. In general, if there are more source fields than target fields, this error is returned.

**LCFAIL_KEY_COUNT_MISMATCH:** Key count mismatch; LEI: <count>, Connector: <count>.

When using key fields, the same number of field names must be provided.

**LCFAIL_STAMPFIELD_TYPE:** Timestamp field <fieldname> must be type datetime; Actual: <type>.

When providing a timestamp field, the field must be of a datetime-compatible data type.

**LCFAIL_FIELD_TYPE:** Type mismatch for field <fieldname>used in this context; Expected: <type>, Actual: <type>.

Certain operations expect certain data types. This error is returned when a field is used for a particular purpose, but does not have the proper data type.

**LCFAIL_MERGE_FIELD:** Field mapping failed due to a missing field <fieldname>.

When mapping fields, a valid match must exist for all required fields. If a field does not have a corresponding match, this error is returned.

**LCFAIL_MISSING_PROPERTY:** No value supplied for required property <property>.

A particular property must be set before the requested operation can be performed. For example, the INDEX property must be provided before attempting to create an index.

**LCFAIL_PROPERTY_CONFLICT:** Conflicting values for properties <property> and <property>.

Values for two properties conflict. In some cases, properties are exclusive or interrelated, and certain combinations are not allowed. See the relevant Connection or Activity documentation for more information.

**LCFAIL_INVALID_PROPERTY:** Invalid property <property>.

A property not supported by the Connection or Activity was referenced.

**LCFAIL_PROPERTY_VALUE:** Invalid property value for property <property>.

A property value is not valid. See the relevant Connection or Activity documentation for more information.

**LCFAIL_INVALID_CHARSET**

The character set indicator is not a valid representation. Valid character set representations are any LEI supported character set suffix, listed in the Supported Character Sets appendix. For example, for LCSTREAMFMT_IBMCP850, use "IBMCP850" (not including the quotes).

**LCFAIL_READ_ONLY_PROPERTY**

An attempt was made to set a read-only property – the Connection does not support assignment of the property, only access.

## Extended Events (Non-Error)

**LCEVENT_CHARACTER_SET:** Unsupported Connection character set <charset>; using native character set.

This status code indicates that a Connection's character set could not be matched to any available LEI character set. This is usually not an error situation, but rather an informational message. The local machine's native character set will be used.

## Extended Fieldname Errors

**LCFAIL_OVERFLOW:** Data overflow in field <fieldname>.

A data overflow occurred when transferring data to or from a field. For numbers, an overflow indicates a value with a magnitude too large. For datetimes, an overflow indicates a year beyond the supported range. For streams, an overflow indicates a value too long. To suppress an overflow error, use the field flag LCFIELDF_TRUNC_DATA or the stream flag LCSTREAMF_TRUNCATE.

**LCFAIL_PRECISION_LOSS:** Data precision loss in field <fieldname>.

Precision loss was detected during type-checking. For efficiency, precision loss is only checked during type-checking, To suppress a precision loss error, use the field flag LCFIELDF_TRUNC_PREC.

**LCFAIL_INVALID_INT:** Invalid integer value in field <fieldname>.

An invalid integer reference, value or text representation was encountered.

**LCFAIL_INVALID_FLOAT:** Invalid float value in field <fieldname>.

An invalid float reference, value or text representation was encountered.

**LCFAIL_INVALID_CURRENCY:** Invalid currency value in field <fieldname>.

An invalid currency reference, value or text representation was encountered.

**LCFAIL_INVALID_NUMERIC:** Invalid numeric value in field <fieldname>.

An invalid numeric reference, value or text representation was encountered.

**LCFAIL_INVALID_DATETIME:** Invalid datetime value in field <fieldname>.

An invalid datetime reference, value or text representation was encountered.

**LCFAIL_INVALID_STREAM:** Invalid stream value in field <fieldname>.

An invalid stream reference or value was encountered.

**LCFAIL_INVALID_FIELD:** Invalid field <fieldname>.

An invalid LCFIELD object instance/handle was used.

**LCFAIL_INVALID_TYPE:** Invalid data type for field <fieldname>.

An invalid type constant was used. Only valid LEITYPE_XXX constants may be used to represent a data type.

**LCFAIL_INVALID_KEY:** Invalid key field <fieldname>.

A key field name provided does not correspond to a valid field. Key field-names must exist in the relevant metadata.

**LCFAIL_DUPLICATE_KEY:** Duplicate key field <fieldname>.

A key field name was used twice in the same key list. Remove the duplicate reference.

**LCFAIL_INVALID_STAMPFIELD:** Invalid timestamp field <fieldname>.

A timestamp field name provided does not correspond to a valid field. timestamp fieldnames must exist in the relevant metadata.

**LCFAIL_INVALID_FIELDNAME:** Field name <fieldname> is not valid in this context.

A field name provided is not valid in the current context. In some cases, field names are restricted in their usage for a particular Connection or Activity.

**LCFAIL_VIRTUAL_FIELD:** Unsupported virtual field <fieldname>.

A virtual field was found, but the field name is not valid for the Connection. See the Connector documentation for a list of valid virtual fields for a particular Connection.

**LCFAIL_VIRTUAL_VALUE:** Invalid data value for virtual field <fieldname>.

Some virtual fields place restrictions on valid field values. See the Connector documentation for additional information on virtual field values for a particular Connection.

**LCFAIL_INVALID_ORDER:** Invalid ordering field <fieldname>

An ordering field name provided does not correspond to a valid field. Ordering fieldnames must exist in the relevant metadata.

## Other General LEI Messages

**Monitor failure -- Error:** This operation requires an active result set, Connector Notes Method – SetProperty.

This message indicates that LEI is attempting to open a document created as a Notes to Notes Real Time activity. However, Notes to Notes Real Time is not supported. Specifically; in a Notes RealTime activity a Notes database cannot be the back-end.

## Activity Logs

### Hidden Fields in Activity Logs

The Activity log documents contain some hidden fields which may be useful. The documentation of these fields is not meant to be a commitment by Lotus to keep this format consistent for the future, and these fields may not be compatible with previous or future LEI versions.

Field “Error”: When an Activity ends in an error state, this field contains a non-zero number that indicates the LEI error code representing the initial error that placed the Activity into an error state. When this field is zero or not present, the activity ended in a non-error state (either successfully, or with all errors explicitly cleared).

Field “CountList”: This is a list of numbers that indicate the counts of internal Connector function calls and record transfers. The entries in this number list are as follows:

Number of calls to Connect

Number of calls to Disconnect

Number of calls to Execute

Number of calls to Select

Number of calls to Fetch

Number of calls to Insert

Number of calls to Update

Number of calls to Remove

Number of calls to Action

Number of calls to Catalog

Number of calls to Create

Number of calls to Drop

*Number of records returned through Execute

*Number of records returned through Select

*Number of records Fetched

*Number of records Inserted

*Number of records Updated

*Number of records Removed

*Number of records returned through Catalog

* - Indicates that when the number count cannot be determined, the value 4294967295 will be used instead, meaning indeterminate.

## Troubleshooting LEI on AS/400

This section addresses some common problems of using LEI on AS/400.

### Some Common Errors and General Hints for Using LEI on AS/400

Below are some common error messages and solutions you may encounter when running an LEI activity.

| *Error in Log* | *Possible Cause* | *Possible Solution* |
|---|---|---|
| Error: Token was not valid. … (-104)<br>Activity Ended With An Error | Non-display characters in SQL statement | Check SQL statement.<br>This error is common if you cut and pasted SQL statement into activity from another source. Note that you can now provide CR/LF characters to improve readability. If SQL statement is correct, try retyping the statement in activity. You may also want to take the DB2 Connector option for providing trace data in the Log Activity. This may pinpoint what is objectionable about the SQL statement. |
| Error: Character conversion cannot be performed. | This message means the LEI job's CCSID is not compatible with the CCSID of the character data in the relational table being reference. For example you could have a job CCSID of 37 and source data CCSID could be 937 (a double byte ccsid). | Consider whether or not you need to run your Domino server under a different LOCALE setting. |
| Error: Authorization failed on distributed database connection attempt. | This message is most likely because the supplied user name is not correct or the provided password is not correct. Note that the password for data sources can be case sensitive. | If going to a remote DB2/400 data source, make sure the password is UPPER case. If going to any other DB2 data source, make sure the password reflects the case sensitivity required on that target platform. |

*continued*

| *Error in Log* | *Possible Cause* | *Possible Solution* |
|---|---|---|
| Error: Relational database <database name> is not in relational database directory. | Either the database name provided in the DB2 connection document is not specified correctly (note no longer case sensitive) or the named database is not registered in the RDB directory. | Do a WRKRDBDIRE CL command and determine if the target database is registered. If not, add the database and the necessary connectivity information to that data source. Note that if you want to connect to the local DB2/400, a named entry for the *LOCAL database must be supplied. |
| Error: <table name> in <library name> not valid for operation. Function -Execute- (-7008) | Most likely reason is that the Journal option is not correct in the Connection or the DB2 table required to be journalized.<br><br>See SQL7008 in the QSQLMSG message file for other possibilities. | If the DB2 table is not journaled on AS/400, verify that the DB2 Connector you are using has non-journaled checked. The DEFAULT setting for all LEI activities is to run with commitment control. Journaling is required to run under commitment control. You must check non-journaled data to turn off this transaction dependency. |

## Finding AS/400 Job Logs

The primary source of information about a failing LEI Server or Activity is the LEI Log database. As LEI is a server addin, you may also need to check the Domino server console for any Domino related diagnostics. Occasionally however, you may need to find an AS/400 system job log. Note that all LEI activities run under the user profile QNOTES on the AS/400. At an AS/400 command line, enter this command to find the job log:

```
WRKSPLF   QNOTES
```

If there is no job log, you may need to change the Job Description for the executing Domino server to generate a job log and then reproduce the failing request. To do this, find the job description associated with the subsystem of your executing Domino server. For example, if your Domino server runs under subsystem DOMINO01, then the job description associated with that Domino server is DOMINO01 (*JOBD) in library QUSRNOTES).

Enter the command:

```
CHGJOBD   QUSRNOTES/DOMINO01   LOG(4 10 *SECLVL)
```

The following are LEI-specific job names:

- LEI
- LEICSMAI
- LEIENGAI

The LEI job controls the overall LEI server, the LEICSMAI job communicates with the LEI control store (the admin databases) and the LEIENGAI job is the process that actually runs the activity. In addition LEI Agents can run under AMgr or HTTP and LEI Real Time runs under SERVER and HTTP.

There is another set of jobs that also factor in to LEI running DB2 Connectioned activities on the AS/400. These are the QSQSRVR jobs. These are pre-started jobs in the QSYSWRK subsystem that perform DB2 access in a thread safe environment on behalf of the activity job (LEIENGAI), the real time job (SERVER or HTTP) or the LEI agent job (AMgr). For each DB2 Connection established by the LEIENGAI, a corresponding QSQSRVR job in the QSYSWRK subsystem will be allocated to that connection and will perform all the DB2 SQL requests on behalf of the Activity job. It may be necessary at times to find a QSQSRVR job log for problem determination. These jobs are pre-allocated under the user profile QUSER. It may be necessary to change the Job Description associated with the QUSER profile in order to generate a job log.

## Additional Tips and Techniques for using LEI on AS/400

Listed below are some general hints when using LEI on AS/400:

- If you are building an Activity that specifies subdirectories within your Domino directory, remember that QNOTES must be authorized to those directories and to use the forward slash (e.g. demodir/mydir/mydoc.nsf)
- Use of the create metadata option when using the DB2 Connection and when the DB2 target table does not exist will result in a CREATE TABLE for the DB2 target (for activities such as direct transfer and replication). A CREATE TABLE done by LEI however, may not be desirable because LEI Text fields are undelimited in size as they are mapped from Notes text fields. When the fields are created in DB2, they can be very, very large. In fact, without further information, a CREATE TABLE will expand all undelimited text fields to fill out the maximum allowable relational record. On AS/400, this is would be character fields that fill out what remains of a 32766 byte record. Although you can qualify the maximum text field size in the Connection Options of the Connection associated with the Notes side of the Activity, that text field size would apply to all undelimited text fields, so it lacks variability. It is strongly suggested you precreate the DB2 table in advance or connect your activity with a Command activity that creates the table with more reasonable CHAR and VARCHAR field lengths. Also note that creating metadata for logging conflicts to DB2/400 can also result in a create table that may not be desirable.
- Do not set your LEI server polling interval too low - 60 seconds is reasonable; 5 or 10 is not.
- It is recommended that for Direct Transfer activities you set the Number of Records to Transfer Concurrently to some number other than 1 for better transfer throughput. Depending on the transfer record size, this might be 20-100.
- If you are using replication with timestamp to replicate a DB2/400 table with a Notes database and your DB2/400 table is restored from backup or has had significant changes applied to it, such as through APYJRNCHG, you should select the Reset Replication Timestamp button prior to the next run of the replication activity. Whenever you suspect that the two databases are significantly out of sync due to a system action or utility you should run with the reset option.

- If you have added a delete trigger to a DB2/400 table for capture deletes on replication, note that this trigger program could get propagated to another object if you do a CRTDUPOBJ on that table. Also note that this trigger program can prevent a FEOD, CLRPFM and CPYF commands from completing successfully. A message indicating the trigger as preventing the request will be logged. These are documented restricted situations when using trigger programs.
- Qualifying a LEI DB2/400 request by specifying a particular member of the physical file is not supported. The first member of the specified physical file (table) is always used.
- If you are taking the overwrite option on Direct Transfer and clearing out the target first, and that target happens to be a DB2/400 table, you might want to consider using a command activity to clear the physical file member first (CLRPFM) and then do the direct transfer without the overwrite option. This would eliminate the potentially lengthy SQL Delete and the additional journal entries that might occur. You could then schedule these two activities as chained (with direct transfer a dependent activity of the clear request).
- If on Replication you receive the message "Update affected no records for Connection, check for key precision mismatch)", you should first try the key precision option on Replication. If this does not improve your results, it could be that your key fields in the generated field list that LEI is working with internally are found to be in a different order than the key fields as specified in the activity "Key Fields" option.

  To work around this situation, you may need to reorder the fields in your form to reflect the order of the key fields as specified in the "Key Fields" option of your Replication activity form or vice versa. Also, you should not use a full timestamp field as a key field. DB2/400 precision is more significant then is Domino's precision when it comes to timestamps and you will not be guaranteed a unique lookup.
- If you get a "Field mapping failed" message and you are providing a list of fields, first verify that the fields specified are correctly identified. If they look fine, then edit the activity document and attempt a copy function over the field list. Occasionally there is an extra, unseen byte that is getting picked up in the list and is causing the mismatch. This can be especially true at the end of the list, or if you are cutting and pasting the list from other locations. Remove the errant byte and try the request again.
- On Direct Transfer, if you specify "Try Update Before Insert", the key field(s) specified in the "Key Fields for Update" field must reflect the source connection, and not the target connection key field names.

- Stored Procedures used on Direct Transfer requests or Real Time activities must run in the activation group of the caller on AS/400 (which is the LEI). To determine the activation group state of your program enter the command DSPPGM. If the Activation group attribute is not *CALLER, you will get an abnormal termination in the log database. In addition, the stored procedure must be invoked under the authority of QUOTES user profile. Make sure that the program is made available to public invocations or that QUOTES is authorized.
- With stored procedures to DB2, also note that NUMBER fields from a Notes form will be bound to a argument values (in C) of type DOUBLE on a call to a stored procedure. Your stored procedure will need to cast accordingly in order to map to the appropriate SQL type.

# Appendix E
# Tutorials

This chapter provides tutorials for building each of the LEI Activities. You can use these tutorials to gain an understanding of how to work with LEI and how to build each LEI Activity.

## Introduction to the Tutorials

The examples in this appendix guide you step-by-step through the process of building LEI Activities, including the Connections that they use. The examples use sample Notes databases supplied with LEI. By following the steps, you should be able to create and run the Activities.

While the examples use an existing Notes database for source data, they leave open the destination databases. The destination can be any LEI Connection you have in your environment and that you choose to use. LEI will create new metadata objects (tables or Notes forms) when transferring or replicating data from the Notes databases to the target databases.

The tutorials in this appendix explore the following tasks:

- **Building a Direct Transfer Activity**

  Build an Activity that transfers data from a sample Notes database to a database of your choice.

- **Building a Polling Activity**

  Build an Activity that polls a sample Notes database for a condition and then triggers a direct transfer Activity.

- **Building a Replication Activity**

  Build an Activity that replicates a sample Notes database to a database of your choice.

- **Building a Scripted Activity**

  Build an Activity to execute LotusScript commands for greater functionality with your data transfers.

- **Building a RealTime Notes Activity**

  Build an a Activity that monitors back-end data in real-time through a Notes application.

  Build an Activity that queries dynamically using your Notes or Web browser.

After building these examples, you will have scheduled Activities that could continue processing. You may want to delete or disable the Activity schedules when you're done with the examples.

## Assumptions

The tutorials assume that you are familiar with Notes and know how to work with Notes. Refer to the Notes online help for assistance if needed.

These tutorials also assume that you have installed LEI, and that you have installed any connectivity software required for your system data sources.

## Before Beginning

Before beginning, you will need to have the Administrator and Log databases on your Notes Client desktop. You will also need to know on which server the LEI sample databases empsamp.nsf and packagetrack.nsf are located. These databases are copied as part of the installation. They will be on the LEI Server machine and will need to be copied to a Notes Server. These sample databases should also be added to your Notes desktop, since during the course of the examples you will modify them.

Be sure that the LEI server is running and can communicate with all data sources that you intend to use. See the *LEI Domino Connectivity and Installation Guide* for information about connectivity requirements and testing connectivity to your data sources.

## Building a Direct Transfer Activity

This section describes the steps in creating a Direct Transfer Activity. This step-by-step process describes the overall procedure for setting up and executing a transfer of data from a Notes database to another database.

The main steps in creating a Direct Transfer Activity are:

1. Create the Connections.
2. Create the Direct Transfer Activity.
3. Run the Activity.
4. View the log.

### Step 1: Create the Connections

Before you create an Activity, you must create the Connections that the Activity will use. A Connection defines access to a database that LEI will interact with. A Direct Transfer Activity requires two Connections: a source database Connection (in this example, the Notes database empsamp.nsf) and a destination database Connection.

**Note** You should have already established connectivity to any database you intend to use.

#### Create the Source Connection

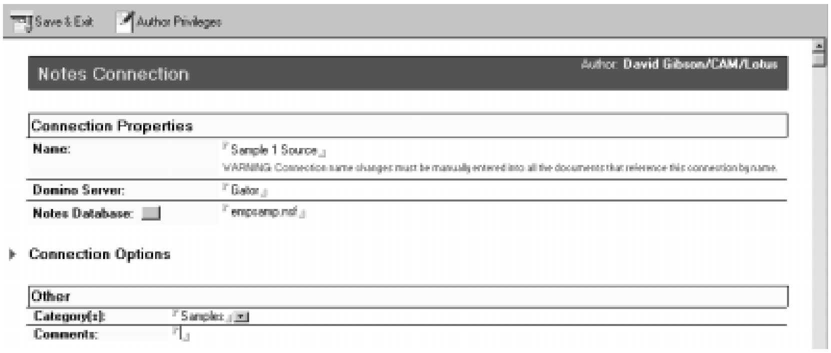
The source Connection is the sample Notes database, empsamp.nsf. To create the source Connection, follow these steps:

1. Start Lotus Notes and open the LEI Administrator database.
2. Choose Create Connection from the LEI Navigator and Notes from the Connectors List.
3. In the Name field, enter a descriptive name that you can use later to identify the Connection from a list. You'll use this name to pick the Connection in the course of creating the Direct Transfer Activity. For the purposes of this example, enter **Sample 1 Source**.
4. Enter the name of the Notes Server on which the Notes database empsamp.nsf is located. Leave this field blank if the database is local.
5. Enter the Notes database name, **empsamp.nsf**.
6. Ignore the Connection Options tab for now.

   This tab allows you to customize the options for handling the data. For information about Connection Options, refer to the chapter in this manual that discusses the associated Connection.

7. Enter the Category, **Samples**. By assigning this category to this and all your sample documents, you will find it easier to manage those documents as a set. This is a practice you may find helpful in practical applications of LEI.

   Your Connection document should look like the following:



Notes Connection

Author: David Gibson/CAM/Lotus

Connection Properties

|  |  |
|---|---|
| Name: | Sample 1 Source<br>WARNING: Connection name changes must be manually entered into all the documents that reference this connection by name. |
| Domino Server: | Gator |
| Notes Database: | empsamp.nsf |

Connection Options

Other

|  |  |
|---|---|
| Category(s): | Samples |
| Comments: |  |

8. Save and exit the document.

### Create the Destination Connection

The Destination Connection document that you create depends on the database that you choose for the destination. To create the Connection, complete the following steps:

1. In the LEI Administrator database, choose Create Connection from the Navigator and then choose the destination database type from the Connectors list.

   This is the database product that you'll use for the other database. It can be any supported database other than Notes.
2. In the Name field, enter **Sample 1 Destination**.
3. Complete the remainder of the document as needed. You must enter the location of the database and may have to enter a user name and password for access. This is the target database in which LEI will create a metadata to receive the data.
4. Enter the Category, **Samples**.
5. Save and exit the document.

Now that you've created the two Connection documents, you can examine the Connections view to make sure that they were created.

### View the Connections Using the Connections View

The Connections view shows you Connections by name or by type. Type refers to the database product type as well as the default view. To view the Connections:

1. In the Administrator, choose Connections from the LEI Navigator.

   To view the Connections By Name, choose View – Connections – By Name from the Notes menu.

2. Find the two Connections you created, **Sample 1 Source and Sample 1 Destination**.

   If you needed to edit those documents, including the Author list, you would double-click the document name in the view and then go into Edit mode (by pressing Ctrl+E or double-clicking within the document).

## Step 2: Create the Direct Transfer Activity

After defining the Source and Target Connections, you can create the Direct Transfer Activity that will transfer data from one database to the other. Complete the steps below to create the Direct Transfer Activity.

1. In the LEI Administrator database, choose Create Activity from the LEI Navigator and Direct Transfer from the Activity list.
2. At Activity Name, enter **Sample Transfer**. This is the name that will identify this Activity.
3. Identify the Source and Target Connections:
   - Use the drop down button for a list of the available Connections or press enter with the cursor in the Connection Name field.
   - From the list of Connections, select **Sample 1 Source**.
   - Repeat for the Connection Name field under Target, selecting **Sample 1 Destination**.
4. In the Form Name field under Source, enter the Notes form **Emps**.
5. In the Form Name field under Target, enter the table name, **Employees**.
6. The Select Statement field specifies the data you want to transfer. The statement can be an SQL query or, for a Notes source database, a selection formula.

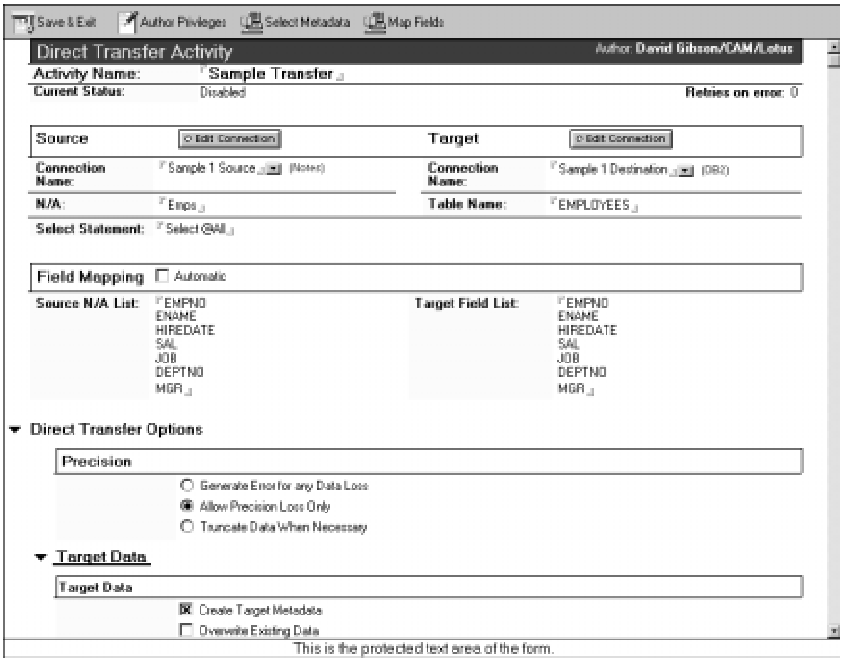
   For this example, enter the following Notes formula: **SELECT @ALL**

   This formula will create a result set identical to that of the set of documents associated with the Notes form **Emps**.

7. Under the Field Mapping Section, choose Automatic and verify that By Field Name is selected.

8. Expand the Direct Transfer Options tab of the document, as shown below.
9. Expand the Target data, then check the Create Target Metadata option to create a new table for the transferred data. Leave the other options as they are.
10. Enter the Category, **Samples**.

    Your Activity should look like the following:



Direct Transfer Activity — Author: David Gibson/CAM/Lotus

Activity Name: Sample Transfer
Current Status: Disabled — Retries on error: 0

| Source | | Target | |
|---|---|---|---|
| Connection Name: | Sample 1 Source (Notes) | Connection Name: | Sample 1 Destination (DB2) |
| N/A: | Emps | Table Name: | EMPLOYEES |
| Select Statement: | Select @All | | |

Field Mapping [ ] Automatic

| | | | |
|---|---|---|---|
| Source N/A List: | EMPNO<br>ENAME<br>HIREDATE<br>SAL<br>JOB<br>DEPTNO<br>MGR | Target Field List: | EMPNO<br>ENAME<br>HIREDATE<br>SAL<br>JOB<br>DEPTNO<br>MGR |

Direct Transfer Options

Precision
- ○ Generate Error for any Data Loss
- ● Allow Precision Loss Only
- ○ Truncate Data When Necessary

Target Data

Target Data
- [x] Create Target Metadata
- [ ] Overwrite Existing Data

This is the protected text area of the form.

11. Save and exit the document.

This completes the process of creating a Direct Transfer Activity.

Because the Activity's schedule was not enabled when you closed the document, you'll have to manually launch it.

## Step 3: Run the Activity

Next, you'll launch the Activity manually and use the Active view to display its status.

To launch the Activity and view the list of running Activities, do the following:

1. In the LEI Administrator, choose Activities from the Navigator. You can also choose View – Activities – By Name from the Notes menu.
2. Locate and select the name of the Activity that you just created, **Sample Transfer**.
3. Click Start Activity from the LEI Navigator.

   You can also choose Action – Run ASAP from the Actions menu or reopen the Activity and use the Run ASAP button. You must have Editor access to the document to use Run ASAP.

### The Active View

The Active view shows the current state of Activities that are either actively scheduled to run or currently running. Check the status of the Activity by opening the Active view.

1. In the LEI Administrator, choose Active from the LEI Navigator.
2. You should see the **Sample Transfer** Activity.

If you do not see the Activity listed, press F9 periodically until it appears as a scheduled job. If you still do not see the Activity listed, it may have already completed. Check the log to see the Activity status.

## Step 4: View the Log

The LEI Log database documents the process and outcome of each Activity.

There are three ways to view the log:

- from the Notes desktop
- from the open Activity document (the Log Link button)
- by choosing Log from the LEI Navigator

**Note** If you open the log from the Activity document, you'll go directly to the Log entry that describes the latest execution of the Activity.

Perform the following steps to open an Activity log using the Notes desktop:

1. Once the Activity has completed, return to the Workspace on your Notes desktop.
2. Select the LEI Log database.
3. Choose Activities by Server from the Navigator.
4. Locate the Sample Transfer Activity and double-click it to display the log.

For more information on logs, see Chapter 4, "LEI Administrator."

## Building a Polling Activity

The steps described here give you the overall procedure for polling. Polling is a way of initiating one or more Activities based on a condition. Once the Polling Activity is running, it checks a selected database at regular specified intervals looking for a particular condition. When the condition returns true, the Polling Activity executes one or more other Activities. The Polling Activity can be thought of as a trigger for other Activities.

In this example, the Polling Activity will trigger a Direct Transfer Activity.

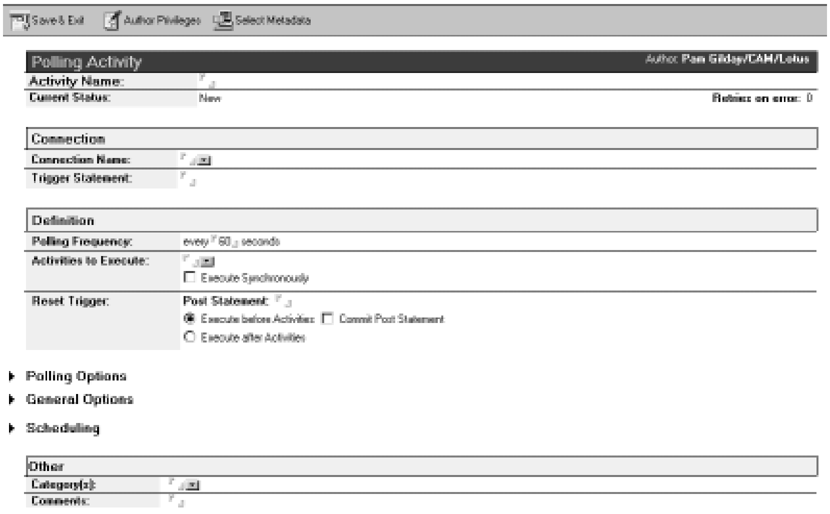
### Step 1: Creating the Polling Activity

The Polling Activity requires a Connection identifying the database to be polled. Instead of creating a new Connection for the purposes of this example, we'll use the existing Connection **Sample 1 Source**.

You'll schedule the Polling Activity to run between 8 AM and 5 PM each day. During that period, it will poll the database once a minute.

Complete the following steps to create the Polling Activity.

1. In the LEI Administrator database, choose Create Activity from the Navigator and then Polling from the Activity list to display the following form.



Polling Activity — Author: Pam Gilday/CAM/Lotus

| | | |
|---|---|---|
| Activity Name: | | |
| Current Status: | New | Retries on error: 0 |

Connection

| | |
|---|---|
| Connection Name: | |
| Trigger Statement: | |

Definition

| | |
|---|---|
| Polling Frequency: | every 60 seconds |
| Activities to Execute: | ☐ Execute Synchronously |
| Reset Trigger: | Post Statement: ● Execute before Activities ☐ Commit Post Statement ○ Execute after Activities |

Polling Options

General Options

Scheduling

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

2. At Activity Name, enter Sample Polling.
3. Identify the database to be polled by entering the Connection name **Sample 1 Source**. Notice that the name completes as you type it.
4. Enter the Trigger Statement **SELECT HIREDATE=@TODAY**. Currently no record satisfies this condition, but you'll add such a record once the Activity is created.
5. At Form Name, enter **Emps**.
6. Leave Polling Frequency unchanged so that the polling occurs every minute. Note that this frequency does not determine when the Polling Activity actually runs. As with other Activities, that is determined by the schedule.
7. At Activities to Execute, select the Activity **Sample Transfer**. This is the Activity that will be executed when the Command Statement returns a true condition (that is, once you add a new record that meets the condition).
8. For this example, leave the Reset Trigger information as it is.

9. Under the Polling Options tab, enter the Maximum Event Count of **1**. The count indicates the maximum number of times that Activities will be executed as a result of a true condition before the Activity terminates. Each true condition and associated execution counts as one event. Zero sets no limit on the number of times that events can occur.
10. Under the General Options tab, if you want to have a specific LEI Server execute the Activity, enter the name of the server in the Designated Server field. Otherwise leave this option blank. Leave other options unchanged.
11. Under the Scheduling tab, change the Schedule field to **SCHEDULE ENABLED**.

    Note that with a Polling Activity, scheduling is slightly different in its effect than with other Activities. Other Activities conclude once they complete their processing. Unless otherwise set up, however, a Polling Activity keeps on running (and polling) once it's initiated. As a result, the Repeat Interval (how often the Activity is initiated) doesn't need to be more frequent than the window of time within which the Activity can run. That window of time, in this case, will be defined in the next step as being from 8 to 5 daily.
12. By default, Repeat Interval is set to every 60 minutes. Change the Repeat Interval to every 1 day.
13. At Run at Times, enter **8:00 AM - 5:00 PM** to have the Activity execute within that span of hours. Given the Repeat Interval of one day, the Activity will start polling at 8:00 AM and stop polling at 5:00 PM. According to the Polling Frequency defined earlier, it will poll every minute within that period of time.

    **Note** If you happen to be doing this example outside that range of hours, use a range that includes the current time.
14. Enter the Category, **Samples**.
15. Save and exit the document.
16. Choose Active from the Navigator to see that the polling Activity Sample Polling is actually in progress but not the Activity **Sample Transfer**.

In the next section, you'll create a record that will cause the polling condition to become true and queue **Sample Transfer** to be run as a result.

### Causing the Polling Condition to be True

To create a true condition and cause the Polling Activity to schedule the subordinate Activity, do the following:

1. On the Notes desktop, select the sample database **EMPSAMP**.
2. Create a new document: choose Create – Emps from the Notes menu.
3. Enter the following information into the form:

| | |
|---|---|
| EMPNO | **7790** |
| ENAME | **NNOTES** |
| JOB | **ENGINEER** |
| MGR | **7791** |
| HIREDATE | **9/12/00** |
| SAL | **1000** |
| COMM | |
| DEPTNO | **72** |

**Note** This document will satisfy the polling condition: **SELECT HIREDATE = @TODAY**.

4. Save and close the form.
5. Choose Active from the Navigator to see that the Polling Activity correctly initiated **Sample Transfer**. Check the log for the status of the Activity.

   **Note** The Polling Activity will continue until it is manually disabled. To stop the Activity, open the Activity and disable the scheduling.

### Cleaning up the Samples

After trying out these examples you can either delete the Connections and Activities or simply disable the Activities, which will allow you to reuse them later on.

To delete the sample Connection and Activity documents, do the following:

1. In the LEI Administrator, choose Categories from the Navigator.
2. The sample documents that you created should be listed under the sub-categories Activity and Connection under the category **Samples**.
3. Select the documents to delete and press the Delete key.
4. The documents will be deleted when you leave the view.

   To inactivate the sample Activity documents, do the following:

5. In the Administrator, choose Categories from the Navigator.
6. The sample documents that you created should be listed under the sub-categories Activity and Connection under the category **Samples**.
7. Select the first Activity document in turn.
8. Press Ctrl+E to edit the document.
9. Under the Scheduling tab, change the Schedule field to **SCHEDULE DISABLED**.
10. Do the same with the other Activities.

## Building a Replication Activity

The steps described here give you the overall procedure for setting up and executing the replication of a database. The databases used are the Notes sample empsamp.nsf and a database of your choice.

The type of replication shown in this example is called "primary key," and is a method of replication that compares one or more key fields in the replicating database and the one to be replicated. By comparing key fields LEI matches records and determines whether replication requires a record update, an insert, or a deletion. An update occurs when the primary keys of both records are identical (assuming the data in the other fields is not), an insert occurs when the primary key exists only in the master, and a deletion when it exists only in the non-master.

### Step 1: Prepare the Database

You'll be using the same databases as in the previous example. By adding a document to the Notes database you will be able to see how the replication works. The record will be replicated to the other database.

1. On the Notes desktop, select the sample database, **empsamp.nsf**.
2. Choose Create – Emps from the Notes menu.
3. Enter the following information into the form:

| | |
|---|---|
| EMPNO | **6627** |
| ENAME | **JNOTES** |
| JOB | **ENGINEER** |
| MGR | **6628** |

|  |  |
|---|---|
| HIREDATE | **Today's date** |
| SAL | **1000** |
| COMM | **0.05** |
| DEPTNO | **70** |

4. Save and close the form.

## Step 2: Create the Connections

Create two Connections: a Notes Connection and a second Connection to a database in your environment. You will identify the master - the one against which the other database is replicated – when you build the Replication Activity.

### First Connection for Replication

This Connection will identify the master database. To create the Connection complete the following steps:

1. In LEI Administrator, choose Create Connection from the Navigator and then Notes from the Connectors List. The Notes Connection document appears.
2. In the Name field, enter **Sample 2 Master**.
3. Enter the name of the Notes Server where the sample database is located.
4. Enter the name of the sample Notes database, **empsamp.nsf**.
5. Enter the Category, **Samples**.
6. Save and exit the document.

### Second Connection for Replication

The next Connection, the "non-master" database, will identify the data that is updated as a result of the replication. To create the Connection:

1. Choose Create Connection from the Navigator and select the database type that you want to use from the Connectors List.
2. At Name, enter **Sample 2 Updated DB**.
3. Enter the information that you entered in the previous example to identify the **Sample 1 Destination** database.
4. Enter the Category, **Samples**.
5. Save and exit the document.

Now that you've created the two Connections, you can check the Connections view to make sure that they we in fact created.

Next, you'll create the Activity that replicates the master database.

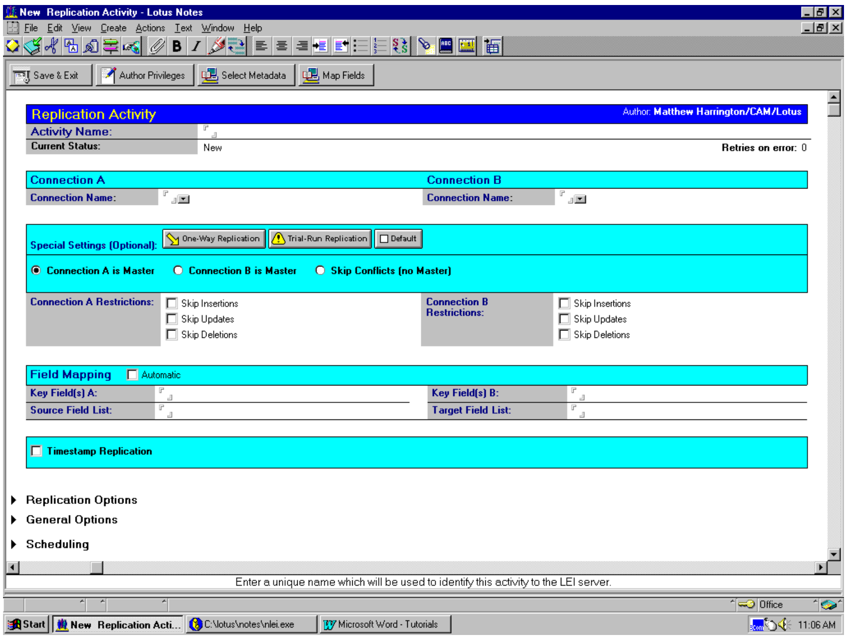
## Step 3: Create the Replication Activity

Once the two Connections are created, create the Activity that will replicate the data. Data is replicated at the table level (or, for Notes, at the form level).

You'll schedule the Activity to run every hour and restrict the window of time within which it can run.

Complete the steps below to create the Replication Activity.

1. In the LEI Administrator, choose Create Activity from the Navigator and then "Replication" from the Activity list. You'll see a replication form open for edit.


New Replication Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields

Replication Activity — Author: Matthew Harrington/CAM/Lotus

| | | |
|---|---|---|
| Activity Name: | | |
| Current Status: | New | Retries on error: 0 |

| Connection A | | Connection B | |
|---|---|---|---|
| Connection Name: | | Connection Name: | |

Special Settings (Optional): One-Way Replication · Trial-Run Replication · ☐ Default

[✓ Connection A is Master] · Connection B is Master · Skip Conflicts (no Master)

| Connection A Restrictions: | | Connection B Restrictions: | |
|---|---|---|---|
| | ☐ Skip Insertions | | ☐ Skip Insertions |
| | ☐ Skip Updates | | ☐ Skip Updates |
| | ☐ Skip Deletions | | ☐ Skip Deletions |

Field Mapping ☐ Automatic

| | | | |
|---|---|---|---|
| Key Field(s) A: | | Key Field(s) B: | |
| Source Field List: | | Target Field List: | |

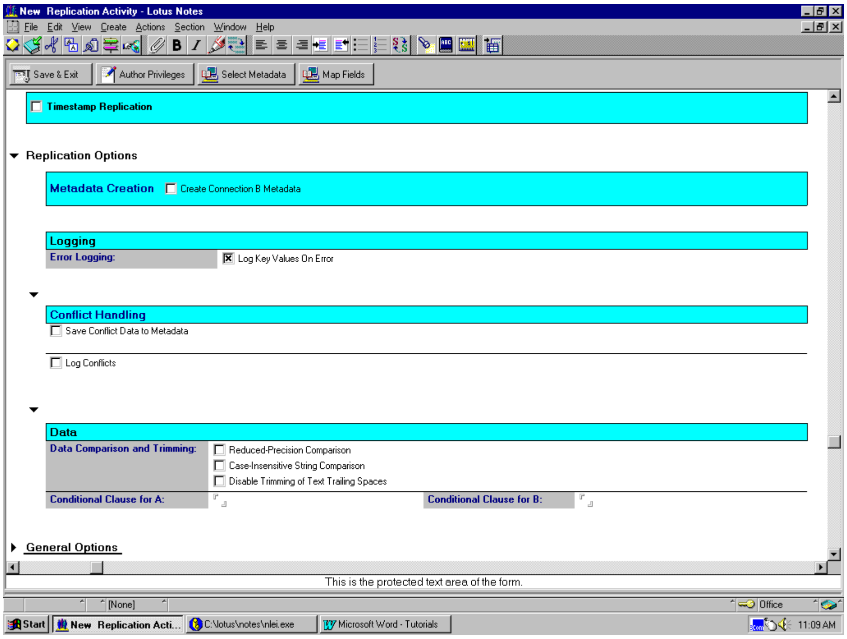
☐ Timestamp Replication

- Replication Options
- General Options
- Scheduling

Enter a unique name which will be used to identify this activity to the LEI server.

2. At Activity Name, enter **Sample Replication**.
3. Select the replication Connections:
   - Under Connection A, bring up the list of Connections using the drop down button or pressing Enter with the cursor in the Connection Name field.
   - From the list of Connections, select **Sample 2 Master**.
   - Do the same thing for the Connection Name field under Connection B, selecting **Sample 2 Updated DB**.

4. Enter the Form Name for both Connections. Enter **Emps** for database A and **Sample1** for database B.
5. In the Special Settings section of the form, choose Connection A is Master. This is the default.
6. Under Field Mapping, identify the primary key fields. Enter **Empno** for both databases A and B. Select Automatic and map fields By Field Name.
7. If the databases that you identify for replication keep the date and time of each change to a record in a timestamp field, you can identify those fields and do primary key/timestamp replication. Leave Timestamp Replication deselected for this example.
8. Expand the Replication Options Tab.


New Replication Activity - Lotus Notes
File Edit View Create Actions Section Window Help
Save & Exit
Author Privileges
Select Metadata
Map Fields
General Options
This is the protected text area of the form.
[None]
Office
Start
New Replication Acti...
C:\lotus\notes\nlei.exe
Microsoft Word - Tutorials
11:09 AM

☐ Timestamp Replication

Replication Options

Metadata Creation ☐ Create Connection B Metadata

Logging
Error Logging: ☒ Log Key Values On Error

Conflict Handling
☐ Save Conflict Data to Metadata
☐ Log Conflicts

Data
Data Comparison and Trimming: ☐ Reduced-Precision Comparison
☐ Case-Insensitive String Comparison
☐ Disable Trimming of Text Trailing Spaces
Conditional Clause for A:
Conditional Clause for B:

For this example, leave the Metadata Creation, Logging, and Conflict Handling sections as they are. Expand the Data section. Select Reduced-Precision Comparison. If the database that you designated as Master uses a different precision than the non-master database for datetime or floating point numbers, false mismatches may occur. This option ensures that this won't happen by using a lower common degree of precision. Datetime values are compared to the second and floating point numbers to ten digits of precision.

9. The Conditional Clause for A could be a Notes formula or SQL WHERE clause to further limit the result set. Leave this field blank for this example.
10. If you want to have a specific LEI Server execute the Activity, select the name of the server at the Designated Server field under the General Options tab. Otherwise leave the General Options as they are.
11. Under the Scheduling tab, check Run Activity Once and Disable. This option only allows the replication to run just once as a trial, after which its schedule is automatically disabled.
12. Change the Schedule field to **SCHEDULE ENABLED**.
13. Limit the times and days on which the Activity can run. For this example, do the following:
    - In the Run at Times field, enter **7:00 AM - 5:00 PM** to have the Activity execute only within that span of hours.
    - Schedule the replication to run every hour. Enter **60** in the Repeat Interval field and select Minutes.
    - In the Days of Week field, delete today’s day name and tomorrow’s by selecting the down arrow and then deselecting the appropriate days of the week.

      In other words, the Activity should not run until 7:00 AM on the day after tomorrow.
14. Enter the Category **Samples**.
15. Save and exit the document.

### Check Activity Status Using the Active View

Check the status of the Activity by opening the Active view

1. In the LEI Administrator, choose Active from the Navigator.
2. You should see the Activity you just built, **Sample Replication**.
3. If the Activity is not listed under the category “Scheduled Jobs”, press F9 periodically until it appears.
4. Note that its next run time is 7:00 AM on the day after tomorrow.
5. Select the Activity.
6. Choose Start Activity from the Navigator.
7. Periodically press F9 in the Active view to refresh the view.

After the Activity completes, it will be scheduled for 7 AM the day after tomorrow. The Run Activity Once and Disable option only counts a scheduled run; not a Run ASAP execution.

## Building a Scripted Activity

This section describes the steps in creating a Scripted Activity. The step-by-step process described here gives you the overall procedure for setting up and running an Activity that will execute an "agent."   An agent can consist of one or more simple actions or formulas to manipulate data in your databases, custom LotusScript commands to perform a number of different functions, or Java statements, each of these possibilities providing greater functionality to your LEI Activities.

This example will show you how to perform a simple action on the data in the database empsamp.nsf using a Scripted Activity.

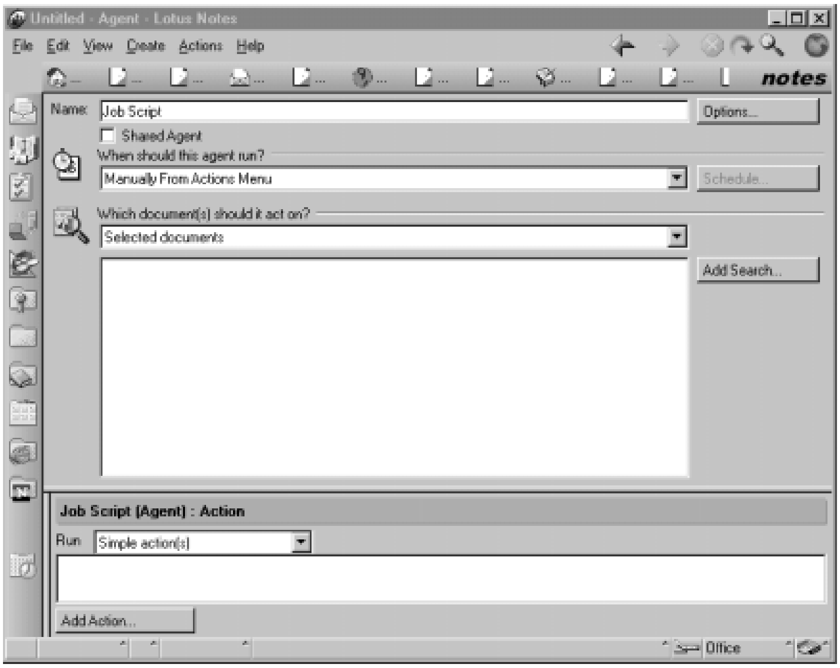
### Step 1: Create the Agent

There are two methods to create an Agent in the database empsamp.nsf.

From within empsamp.nsf you can choose Create – Agent… from the Notes menu or you can choose the Create Agent button when creating the Scripted Activity. For this example, we will create the Agent when creating the Activity.

1. In the LEI Administrator, choose Create Activity from the Navigator and then "Scripted" from the Activity list.
2. In Activity Name, enter **Scripted Activity**.
3. Click the Create Agent button at the top of the document.
4. A dialog box will prompt you for a server name. Enter the server name where the database empsamp.nsf is located.
5. A dialog box will then prompt you for a database name in which to create the agent. Enter **empsamp.nsf**.

6. An Untitled Agent Window opens, as seen below.


Untitled - Agent - Lotus Notes
File Edit View Create Actions Help
notes
Options...
Schedule...
Add Search...
Add Action...
Office

Name: Job Script

Shared Agent

When should this agent run? Manually From Actions Menu

Which document(s) should it act on? Selected documents

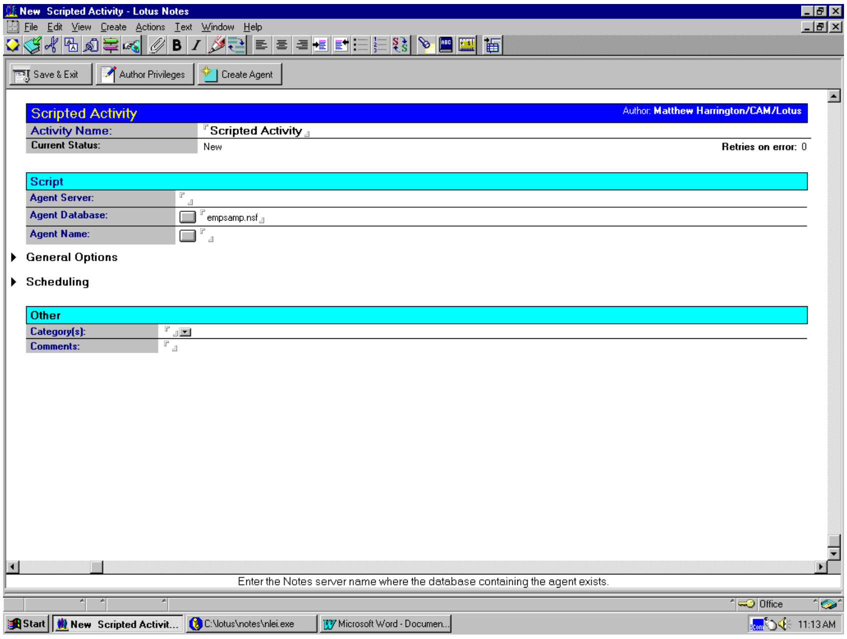
Job Script (Agent) : Action

Run: Simple action(s)

7. Enter **Job Script** as the Name of the Agent.
8. Under "When should this agent run?" select Manually From Actions Menu.
9. Under "Which document(s) should it act on?" select All documents in database.
10. For the Job Script (Agent): Action choice, choose Simple action(s).
11. Click the Add Action button at the bottom of the window. An Add Action box will appear. In the Action field, select "Modify Field." In the *Field* field, select the "JOB" field. In the *Value* field, enter **ENGINEER** and select *Replace Value*. This action will replace the value in the JOB field for each document in the empsamp.nsf database. Click OK.
12. Save and close the Agent. Notice that you are left in the empsamp.nsf database.

## Step 2: Complete the Scripted Activity

Press Escape to return to the Scripted Activity that you created.


New Scripted Activity - Lotus Notes
File Edit View Create Actions Text Window Help
Save & Exit
Author Privileges
Create Agent

Scripted Activity — Author: Matthew Harrington/CAM/Lotus

| | |
|---|---|
| Activity Name: | Scripted Activity |
| Current Status: | New — Retries on error: 0 |

Script

| | |
|---|---|
| Agent Server: | |
| Agent Database: | empsamp.nsf |
| Agent Name: | |

General Options

Scheduling

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

Enter the Notes server name where the database containing the agent exists.

Office
Start
New Scripted Activit...
C:\lotus\notes\nlei.exe
Microsoft Word - Documen...
11:13 AM

1. Click the button in the Agent Name field and choose Job Script from the list of Agents.
2. If you want to have a specific LEI Server execute the Activity, select the name of the server at the Designated Server field under the General Options tab. Otherwise leave the General Options as they are.
3. Enter the Category, Samples.
4. Save and exit the document.

## Step 3: Launch the Activity

To launch the Activity and view it, complete the following steps.

1. In the LEI Administrator, choose Activities. Alternatively, you can choose View – Activities – By Name.
2. Locate and select the Activity that you just created: Scripted Activity.
3. Choose Start Activity from the Navigator.
4. Choose Active from the Navigator.
5. You should see the Activity you just built, Scripted Activity, on the list under the category Scheduled Jobs. Periodically press F9 to refresh the view and see the current disposition of the Activity.

### Step 4: View the Log and the Results of the Activity

Go to the log and find the Activity Scripted Activity and double-click it to see the log. The Activity should be logged as “Started” and “Finished.”

Open the empsamp.nsf database and take note the Job field. Each instance has been replaced with the value “ENGINEER”.

This completes a simple Scripted Activity.

## Building a RealTime Notes Activity

The most common use for RealTime is to give access to back-end databases such as DB2, Oracle, Sybase, to Notes users. This section provides an example of how to give Notes users information about employees, where the employee information is stored in a back-end database and might look something like the empsamp.nsf sample database supplied with LEI. The EMPNO key field and the ENAME, HIREDATE, SAL, JOB, and DEPTNO data fields are used.

You can perform the steps in this example yourself by transferring this Notes data to a back-end database, if you have not already done so, using an LEI Direct Transfer Activity. the process for creating an LEI Direct Transfer Activity was described earlier.

**Note** To operate the RealTime Activities, the LEI server must be installed on the same Domino server as the Notes database that Notes end-users will access for RealTime lookups.

Follow these steps to create a simple RealTime application.

### Step 1: Gather the Necessary Information

Identify the table name, description (metadata) and necessary user name and password in the back-end database. If you followed the previous Direct Transfer example, the table name is Employees, the field names are as above, and you will know the user name and password needed for the back-end database.

### Step 2: Create the Form in the Notes Database

Create a form in the Notes database having corresponding fields for the key(s) and data fields. The field names do not need to match those used in the back-end database, however the datatypes should match.

Enter a form name of your choice.

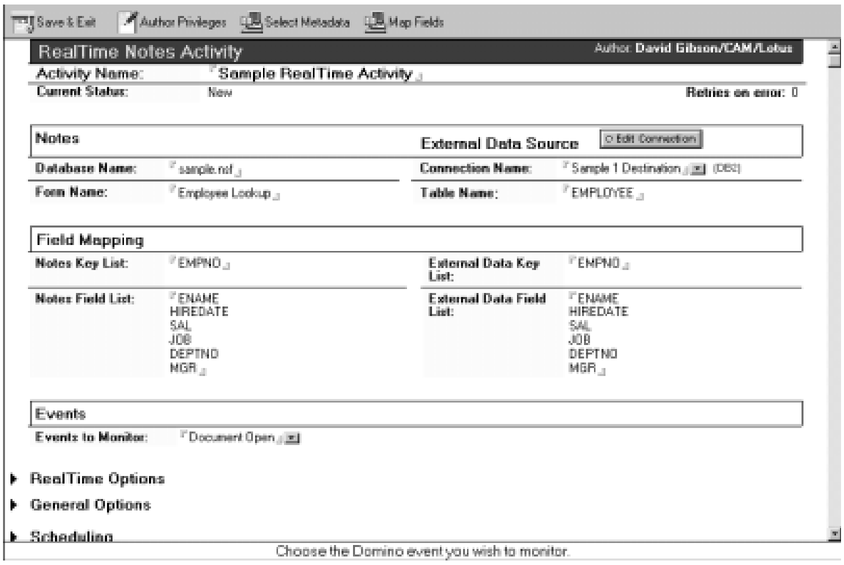
Create a view to select documents from later. For this example, use the view name “Employee Lookup”.

## Step 3: Create the Connection Document to the Back-end Database

Create a Connection document to the back-end database using the information gathered in step 1 above, or reuse one that you might have created in a previous example , Sample 1 Destination.

## Step 4: Create the RealTime Activity in the LEI Administrator

Create the RealTime Activity in the LEI Administrator using the above Connectors. Map the key and data fields that correspond. This RealTime Notes Activity example uses the “Document Open” Monitor, which causes a lookup against the DB2 table EMPLOYEE when end-users open the form EMPS with only the keyfield EMPNO populated.



RealTime Notes Activity — Author: David Gibson/CAM/Lotus

Activity Name: Sample RealTime Activity
Current Status: New — Retries on error: 0

**Notes** — **External Data Source** [Edit Connection]

| | | | |
|---|---|---|---|
| Database Name: | sample.nsf | Connection Name: | Sample 1 Destination (DB2) |
| Form Name: | Employee Lookup | Table Name: | EMPLOYEE |

**Field Mapping**

| | | | |
|---|---|---|---|
| Notes Key List: | EMPNO | External Data Key List: | EMPNO |
| Notes Field List: | ENAME<br>HIREDATE<br>SAL<br>JOB<br>DEPTNO<br>MGR | External Data Field List: | ENAME<br>HIREDATE<br>SAL<br>JOB<br>DEPTNO<br>MGR |

**Events**

Events to Monitor: Document Open

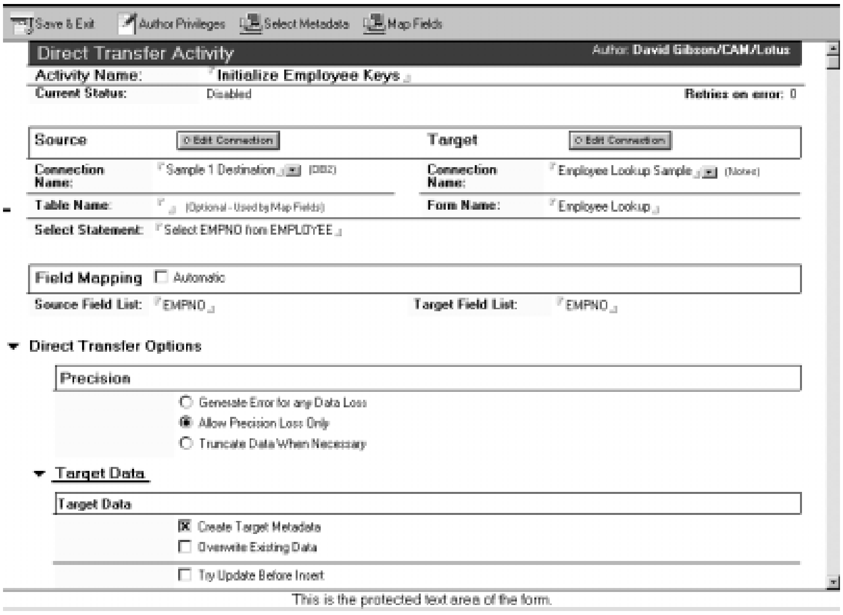
Choose the Domino event you wish to monitor.

## Step 5: Initialize the Keys in the Notes Database

The Notes database above must be initialized by creating document “stubs” which contain the keys that refer to the records in the back-end database. One way to do this is to build an LEI Direct Transfer Activity to create documents in Notes using the form created in step 2. Populate with only the key values.

See the section entitled “Building a Direct Transfer Activity” in this appendix for more information.

An example is shown below.



Direct Transfer Activity

Author: David Gibson/CAM/Lotus

| | | | |
|---|---|---|---|
| Activity Name: | Initialize Employee Keys | | |
| Current Status: | Disabled | Retries on error: | 0 |
| **Source** | | **Target** | |
| Connection Name: | Sample 1 Destination (DB2) | Connection Name: | Employee Lookup Sample (Notes) |
| Table Name: | (Optional - Used by Map Fields) | Form Name: | Employee Lookup |
| Select Statement: | Select EMPNO from EMPLOYEE | | |

Field Mapping ☐ Automatic

Source Field List: EMPNO    Target Field List: EMPNO

Direct Transfer Options

Precision

○ Generate Error for any Data Loss
● Allow Precision Loss Only
○ Truncate Data When Necessary

Target Data

Target Data

☒ Create Target Metadata
☐ Overwrite Existing Data

☐ Try Update Before Insert

This is the protected text area of the form.

## Step 6: Start the RealTime Activity

1. Return to the LEI Administrator database and use the Activities view to locate your Activity.
2. Either choose Start Activity or open the Activity and establish an interval where the RealTime Notes activity should be operational for Notes end-users.
3. Enable the schedule, and save the document.

   **Note** The Activity will continue to run until you either stop it by choosing Stop Activity or the schedule expires.
4. Refresh the Activities view until the Activity status icon become a beacon, indicating that the activity is now running.

## Step 7: Use the RealTime Activity

Use the application by opening the monitored Notes database and opening one of the documents containing a key value.

The first open will take longer because the Connection is established with the back-end data base. The data fields should then be populated by RealTime from the back-end database.

This completes the example of using a RealTime Notes activity.

## Building a Dynamic RealTime Query for Notes or Web Clients

This section provides information about using your Notes or Web browser client to invoke a RealTime Notes Activity that accepts a key value input by the user for querying an external data source. This extends the functionality of the RealTime Notes Activity to provide a dynamic query capability.

### Example Contents and Setup

Included with your LEI installation disk is a sample database packagetrack.nsf. This database provides the Lookup and Status forms used in the example. The database also includes sample raw data that may be used to populate your back-end database for uses with the example.

### Overview

There are several pieces needed to create a RealTime Dynamic Notes or Web based queries:

- A back-end database containing the information that you want to present to users in Notes,
- A Notes database on your Domino server, such as this packagetrack.nsf.
- An LEI or DECS RealTime Activity which resides on the same Domino server and connects the two.

To follow this example you will move the sample data to the back-end database, create a two forms and a view in your Notes database and then a RealTime Activity with LEI to monitor the Notes database. This example database has the necessary forms and views if you do not wish to create them yourself.

### Step 1: Setup

Copy the **packagetrack.nsf** database to a directory on your Domino server where LEI is installed and RealTime activated.

**Note** If you wish to modify the forms, you must open the packagetrack.nsf in Domino Designer to gain access to the forms.

### Step 2: Create or Choose a Back-end Database to Monitor

Within the packagetrack.nsf database there is data on packages that can be transferred to your back-end database. These documents or records have been created with the Package form and can be viewed with the Packages view. If you are using LEI you can move this data to your back-end database, or, if not, choose another similar table of your own or create the table. The key field “CODE” is important to us here. If you are using a table of your own, identify the key field you wish to use for lookups and use it in the appropriate places in the later steps. We will use PackageID on the Notes side to illustrate that the field names need not be the same in both databases.

**Note** If you are using an LEI Direct Transfer Activity to move this data to Oracle you should be careful to select the option for “No Long Column” in the Connection document. Otherwise the first text column will be made a LONG in Oracle.

## Step 3: Create the Lookup Form in the Notes Database

If you are following this example you can use the form “Lookup” included in this database. The key field which will be used by LEI for the lookup is the important item here. In this example it is PackageID. You can put information on this form to guide the user.

**Note** There is a view “Query” referred to here that you will need to create after the form is completed.

There is a button in the form for “Lookup Package” that contains the following formula:

```
@PostedCommand ([FileSave]);

@PostedCommand ([FileCloseWindow]);

@PostedCommand ([FileOpenDatabase]; @DbName; "Query";
@Text(PackageID));

@PostedCommand ([OpenDocument])
```

This code is used when running the application from Notes but is ignored when running the application from the Web.

**Note** The “document create” event should not be monitored when performing dynamic queries. This can lead to an unwanted record being inserted in the back-end or the error “duplicate key value specified”.

### Explanation

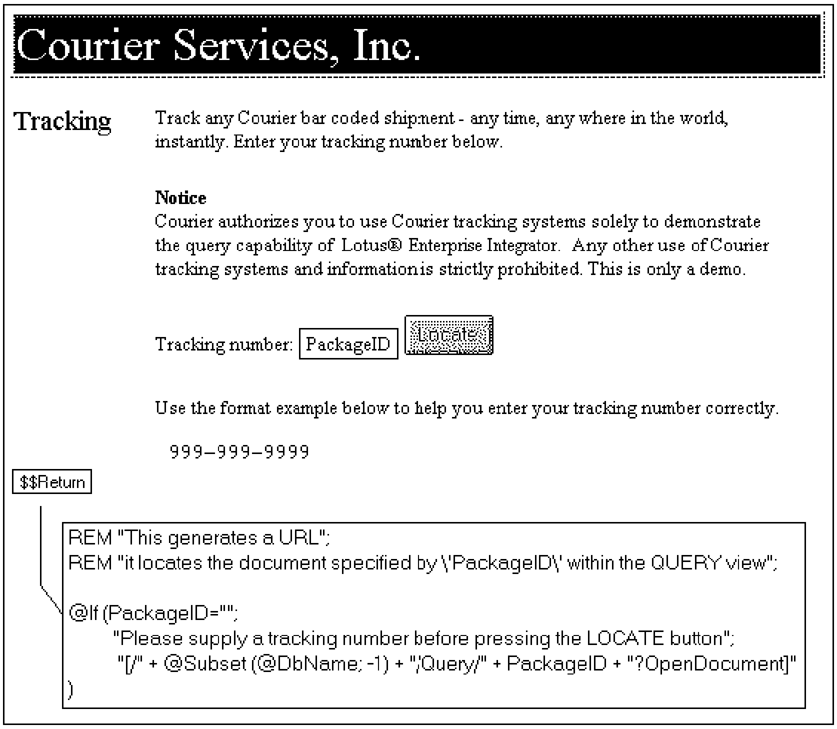
1. Save the document with the PackageID that the user wants entered into that field.
2. Close the window.

   **Note** If you do not close the window, an orphaned “Untitled” is left.
3. Locate the document in the view “Query” where the indexed key field PackageID is the same as the one just saved.
4. Open the document. The view selection formula will then display the Status form.

From the Web, Domino will treat the button as a POST command and will create a document in the database and then respond to the formula in the $$Return field, which should be hidden. The $$Return field will generate a URL that will open the newly created document in a view called Query just as it happens with the Notes client.

The $$Return field contains the following:

```
@If (PackageID="";
         "Please supply a tracking number before pressing the
LOCATE button";
          "[/" + @Subset (@DbName; -1) + "/Query/" +
PackageID + "?OpenDocument]"
```



Courier Services, Inc.

Tracking

Track any Courier bar coded shipment - any time, any where in the world, instantly. Enter your tracking number below.

**Notice**
Courier authorizes you to use Courier tracking systems solely to demonstrate the query capability of Lotus® Enterprise Integrator. Any other use of Courier tracking systems and informationis strictly prohibited. This is only a demo.

Tracking number: PackageID Locate

Use the format example below to help you enter your tracking number correctly.

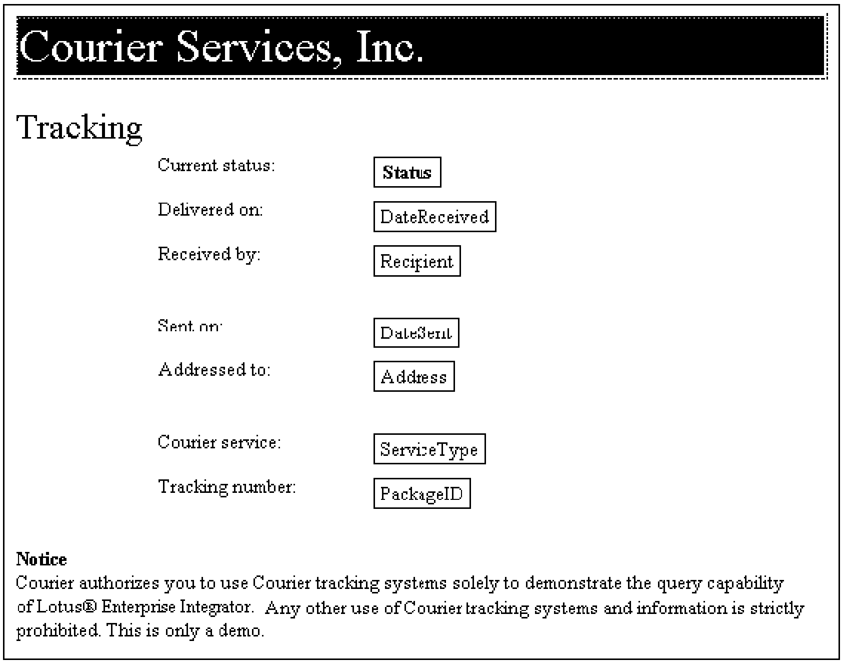
999-999-9999

$$Return

```
REM "This generates a URL";
REM "it locates the document specified by \'PackageID\' within the QUERY view";

@If (PackageID="";
      "Please supply a tracking number before pressing the LOCATE button";
       "[/" + @Subset (@DbName; -1) + "/Query/" + PackageID + "?OpenDocument]"
)
```

## Step 4: Create the Status Form in the Notes Database

Create another form to return the information to your user. In our example it is called Status. This form must also use the key field “PackageID”. The RealTime Activity will be asked to “Monitor any form” so that it will respond to the opening of this form when the original document created by the user is opened from the view, “Query”. By using a Form Filter formula only the two forms of interest will trigger the RealTime Activity.



Courier Services, Inc.

Tracking

| | |
|---|---|
| Current status: | Status |
| Delivered on: | DateReceived |
| Received by: | Recipient |
| Sent on: | DateSent |
| Addressed to: | Address |
| Courier service: | ServiceType |
| Tracking number: | PackageID |

**Notice**
Courier authorizes you to use Courier tracking systems solely to demonstrate the query capability of Lotus® Enterprise Integrator. Any other use of Courier tracking systems and information is strictly prohibited. This is only a demo.

## Step 5: Create the Query View in the Notes Database

We will need a view to find the document we just created referenced in the last step. This view must be indexed on the key field, in our example PackageID.

The view selection formula is:

```
SELECT (@Contains(Form; "Lookup")).
```

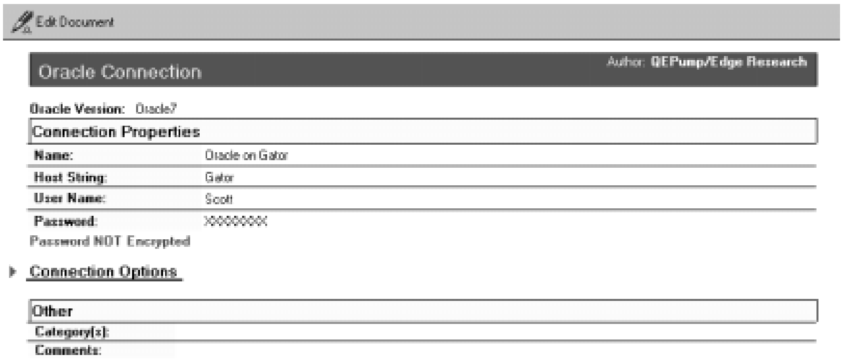
Under the View Properties Advanced tab, put the following formula:

```
@If(@IsNewDoc;"Lookup"; "Status").
```

This will switch the form used to the Status form when opening a document that has been saved using the Lookup form.

## Step 6: Create the Connectors to the Source and Back-end Databases

We will need an LEI Connector to our back-end database similar to the following:



Oracle Connection

Author: QEPump/Edge Research

Oracle Version: Oracle7

Connection Properties

| | |
|---|---|
| Name: | Oracle on Gator |
| Host String: | Gator |
| User Name: | Scott |
| Password: | XXXXXXXX |

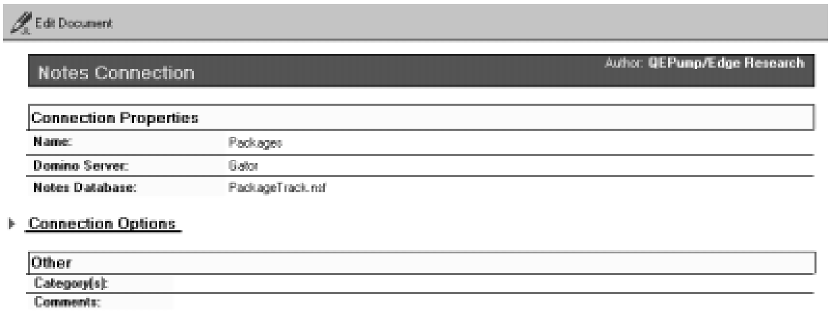
Password NOT Encrypted

Connection Options

Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

And one to the Notes database being monitored by RealTime.



Notes Connection

Author: QEPump/Edge Research

Connection Properties

| | |
|---|---|
| Name: | Packages |
| Domino Server: | Gator |
| Notes Database: | PackageTrack.nsf |

Connection Options

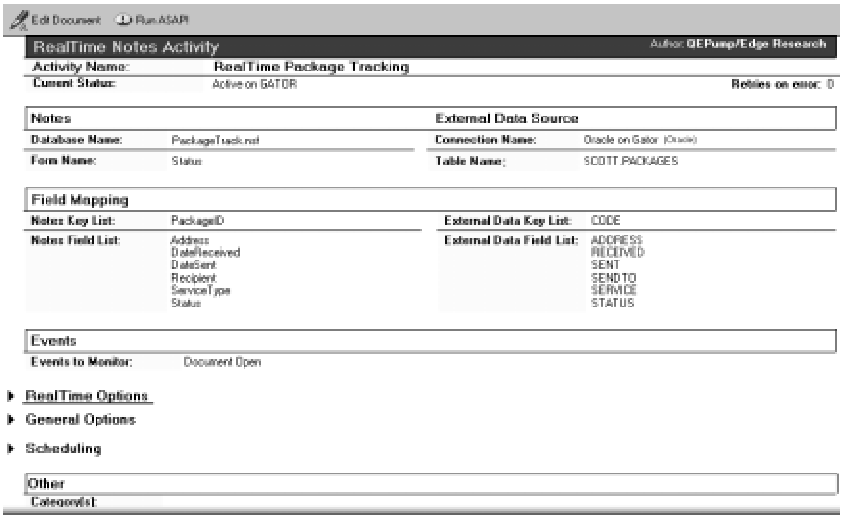
Other

| | |
|---|---|
| Category(s): | |
| Comments: | |

## Step 7: Define the RealTime Notes Activity

Create the RealTime Activity in the LEI Administrator that will monitor the Notes database for Open events. It will use the Connector created above to link the back-end database to the Notes document using the key field you have chosen. In this case the key field in Notes is PackageID and in the back-end database is CODE.

Choose the form Status so that the data fields are available for to map. We need to monitor the Lookup form in order to catch its Open event so we must check “Monitor All Forms”. However in our sample database there are other forms that are unrelated to the functioning of this application so we want to exclude them by using a filter formula FORM = “Status” ! “Lookup”.



RealTime Notes Activity — Author: QEPump/Edge Research

Activity Name: RealTime Package Tracking
Current Status: Active on GATOR — Retries on error: 0

**Notes**
Database Name: PackageTrack.nsf
Form Name: Status

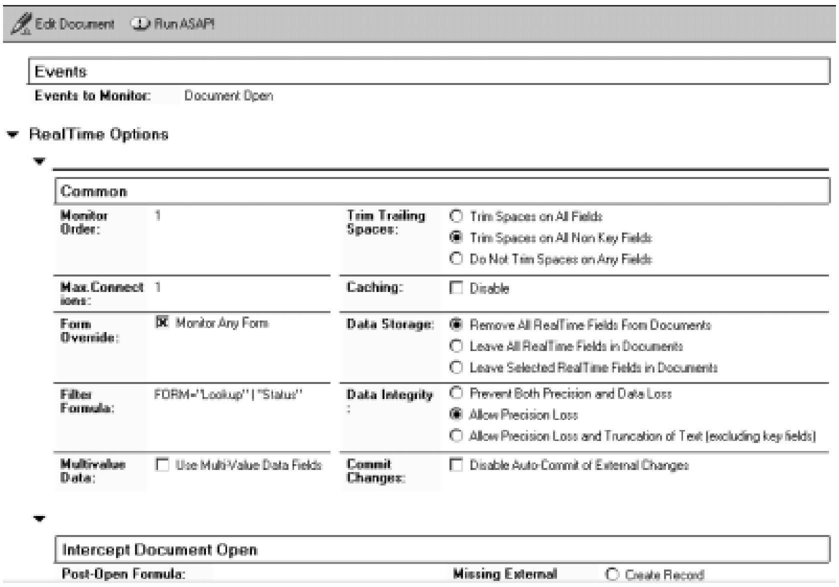
**External Data Source**
Connection Name: Oracle on Gator (Oracle)
Table Name: SCOTT.PACKAGES

**Field Mapping**
Notes Key List: PackageID
Notes Field List: Address, DateReceived, DateSent, Recipient, ServiceType, Status
External Data Key List: CODE
External Data Field List: ADDRESS, RECEIVED, SENT, SENDTO, SERVICE, STATUS

**Events**
Events to Monitor: Document Open

RealTime Options
General Options
Scheduling

**Other**
Category(s):



Events

Events to Monitor: Document Open

RealTime Options

Common

| | | | |
|---|---|---|---|
| Monitor Order: | 1 | Trim Trailing Spaces: | ○ Trim Spaces on All Fields<br>● Trim Spaces on All Non Key Fields<br>○ Do Not Trim Spaces on Any Fields |
| Max.Connections: | 1 | Caching: | ☐ Disable |
| Form Override: | ☒ Monitor Any Form | Data Storage: | ● Remove All RealTime Fields From Documents<br>○ Leave All RealTime Fields in Documents<br>○ Leave Selected RealTime Fields in Documents |
| Filter Formula: | FORM="Lookup" \| "Status" | Data Integrity: | ○ Prevent Both Precision and Data Loss<br>● Allow Precision Loss<br>○ Allow Precision Loss and Truncation of Text (excluding key fields) |
| Multivalue Data: | ☐ Use Multi-Value Data Fields | Commit Changes: | ☐ Disable Auto-Commit of External Changes |

Intercept Document Open

Post-Open Formula: Missing External ○ Create Record

## Step 8: Final Steps

1. Ensure that you have populated your back-end source with the packagetrack.nsf sample data as mentioned earlier.
2. Enable (schedule) the RealTime Notes Activity you have created if you want to make it available for a specific interval of time or run ASAP to have it active until you close it manually.
3. Verify that the Activity is active by refreshing the view and watching for the Active icon. If the Error icon appears, you must go back and check your work.
4. From a Notes client create a Lookup document from the Actions menu and type in a tracking number in the PackageID field. Refer to the packagetrack.nsf database for example numbers.
5. Click the Locate button to save the Lookup document in the database.

   The same document is reopened in the view. The selection formula in the view then switches the form to Status when it sees that the document has been previously saved.
6. From a Web browser access the Lookup form using a URL similar to this:

   ```
   http://Gator/packagetrack.nsf/Lookup?OpenForm
   ```

   **Note** You will need to modify this to reflect the name of your own Domino server and any other changes you might have made in the names used in your application.

7. Type in a tracking number (PackageID field). Refer to the packagetrack.nsf database for example numbers.
8. Click the Locate button.

When you click the Locate button, the Lookup document is saved in the database and the $$Return text is passed to the Domino HTTP process. The new document is reopened. The RealTime Notes Activity is activated, takes the value submitted to the Lookup form, and maps it to the RealTime Notes Activity definitions and Status form. The PackageID value is then sent to the back-end source and the table records are searched according to the PackageID value. Results of the search are sent back to the Web application as directed by the URL specified in the Lookup form.

# Index

Lotus®

*An IBM Company*

SUPERHUMANSOFTWARE™

www.lotus.com

e-business

Part No. CT7KFNA